VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Simulace modulace a demodulace digitálních signálů pomocí programu GNU Octave

Simulation of Modulation and Demodulation of Digital Signals with GNU Octave

2015 Lukáš Janča

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Lukáš Janča**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: Simulace modulace a demodulace digitálních signálů pomocí programu GNU Octave
Simulation of Modulation and Demodulation of Digital Signals with GNU Octave

Zásady pro vypracování:

Cílem bakalářské práce je simulace ASK, FSK, PSK a QAM modulace a demodulace digitálních signálů s využitím programu GNU Octave.
1. Popis modulací a demodulací digitálních signálů.
2. Senámení s programem Octave.
3. Simulace jednotlivých typů modulace.
4. Vytvoření návodů pro počítačová cvičení.

Seznam doporučené odborné literatury:

Lajos L. Hanzo, Soon Xin Ng, Thomas Keller, William Webb, *Quadrature Amplitude Modulation: From Basics to Adaptive Trellis-Coded, Turbo-Equalised and Space-Time Coded OFDM, CDMA and MC-CDMA Systems* Wiley-IEEE Press 2004, ISBN-13: 978-0470094686
Mathuranathan Viswanathan, *SIMULATION OF DIGITAL COMMUNICATION SYSTEMS USING MATLAB* Amazon Digital Services, ISBN-13: 978-1301525089
Podle pokynů vedoucího bakalářské práce

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Nevlud**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1 OSTRAVA

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Rád bych poděkoval vedoucímu práce Ing. Pavlu Nevludovi za odbornou pomoc a konzultaci, kterou mi poskytl při vytváření této bakalářské práce.

Abstrakt

Bakalářská práce se zabývá tvorbou simulace pro modulaci a demodulaci digitálního signálu pomocí programu GNU Octave. Konkrétně se jedná o digitální modulace typu ASK, FSK, PSK a QAM. Součástí teoretického výkladu je vysvětlení principů modulace a demodulace, objasnění hlavních pojmů a rozdělení základních signálů a modulací.
Základní vlastnosti, průběh instalace, nutná konfigurace, výhody a nevýhody programu GNU Octave jsou zmíněny.
Samotná realizace simulace je doplněna programovým kódem a grafickým výstupem. Funkčnost použitých příkazů a vlastních algoritmů je vysvětlena. V rámci simulace se pracuje i s šumem jako AWGN.
Bližší vysvětlení vykreslovacích funkcí je součástí tvorby návodů pro cvičení, kde se taktéž rozvádí automatizace simulace. Celkový programový kód je umístěn na přiloženém DVD pro volné šíření a použití v programu GNU Octave.

Klíčová slova: Digitální modulace, demodulace, simulace, ASK, FKS, PSK, QAM, AWGN, GNU Octave.

Abstract

Bachelor's thesis deals with simulation of modulation and demodulation of digital signal with use of GNU Octave software. Concretely it's about digital modulation for ASK, FKS, PSK and QAM. Included in theoretical exposition is explanation of modulation and demodulation processes, main terms and dispensation of basic signals and modulations.
Basic properties, process of installation, needed configuration, advantages and disadvantages of program GNU Octave are mentioned.
Realization of simulation itself is made of program code and graphic output. Functionality of used commands and personal algorithms is explained. Processing of noise like AWGN is within the simulation.
Closer explanation of drawing functions is part of making a tutorials for school lessons, where the automatization of simulation is done. Overall program code is placed on the DVD for free spreading and use on the GNU Octave software.

Keywords: Digital modulation, demodulation, simulation, ASK, FSK, PSK, QAM, AWGN, GNU Octave.

Seznam použitých zkratek a symbolů

AM	–	Amplitude modulation – Amplitudová modulace
ASK	–	Amplitude-shift keying – Klíčování amplitudovým zdvihem
AWGN	–	Additive White Gaussian Noise -– aditivní bílý Gaussovský šum
FM	–	Frequency modulation – Frekvenční modulace
FSK	–	Amplitude-shift keying – Klíčování frekvenčním zdvihem
GNU	–	GNU's Not UNIX – GNU Není UNIX!
GNU GPL	–	GNU General Public License – všeobecná veřejná licence GNU
PM	–	Phase modulation – Fázová modulace
PSK	–	Phase-shift keying – Klíčování fázovým zdvihem
QAM	–	Quadrature amplitude modulation – Kvadraturní amplitudová modulace
SNR	–	Signal-to-noise ratio – poměr výkonu signálu a výkonu šumu
A	–	Značka amplitudy - jednotka například napětí
dB	–	Jednotka decibelu – měřítko podílu dvou hodnot
Hz	–	Jednotka hertze – veličina frekvence
t	–	Značka času - jednotka vteřina
φ_0	–	Značka počáteční fáze – jednotka rad nebo $^\circ$
ω	–	Značka úhlového kmitočtu – jednotka Hz
$^\circ$	–	Jednotka úhlového stupně – míra rovinného úhlu
rad	–	Jednotka radiánu – jednotkový úhel v obloukové míře

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cílem této práce je vytvořit simulace pro konkrétní typy modulací a demodulací digitálního signálu s využitím programu GNU Octave. Strukturu vypracované práce jsem rozdělil do čtyř hlavních kapitol.

V první kapitole se věnuji teoretickému výkladu, ve kterém jsou popsány principy procesu modulace a demodulace, základní pojmy k teorii digitálních signálů a rozdělení konkrétních typů modulací, které jsou cílem této práce.

Navazující kapitola se týká seznámení s použitým programem GNU Octave. Jedná se o výpis základních vlastností programu, primárního uzpůsobení programu a nabídky souboru nástrojů či balíčků funkcí. Dále popisuji průběh instalace programu a nutnou konfiguraci, kterou jsem před prvním spuštěním provedl.

Ve třetí kapitole řeším realizaci simulace, která je doplněna programovým kódem a grafickým výstupem jednotlivých úloh. Simulovány jsou jednoduché a složené typy modulací. K těmto namodulovaným signálům je poté přidán šum a jsou následně demodulovány.

Za účelem vytvořit návody pro cvičení se v poslední kapitole více podrobněji zabývám vysvětlením vykreslovací funkce. Na závěr uvádím příklady pro výpis do příkazového řádko a řešení, jak zautomatizovat kód pro minimální nutnou úpravu kódu a snažší simulaci a způsoby konzolového výstupu.

2 Popis modulací a demodulací digitálních signálů

2.1 Definice hlavních pojmů

Záměr člověka dorozumět se (přenášet určitou informaci) na větší vzdálenosti dal vzniknout oboru, který dnes nazýváme telekomunikace. Pojem informace je v telekomunikacích vyjádřen formou zprávy, která je vytvořena určitým zdrojem zpráv. Každá zpráva se skládá z jednotlivých prvků (např. písmena, číslice, apod.). Tyto prvky tvoří tzv. abecedu zdroje zpráv. Abecedu označujeme za omezenou, zdali obsahuje pouze konečný počet prvků a za abecedu neomezenou, pokud má počet prvků neomezený.

V případě, když máme k dispozici neomezený počet prvků, jsme schopni vytvořit spojitou zprávu (analogovou). V opačném přídadě můžeme vytvořit pouze nespojitou zprávu (diskrétní).

Již určitou fyzikální formu zprávy nazýváme signál. Signály lze rozdělit na:

- Signál spojitý (analogový) – vyjadřuje zprávu pomocí neomezeného počtu hodnot určité fyzikální veličiny (např. napětí).
- Signál nespojitý (diskrétní) – je nespojitý bud' v čase, v amplitudě, případně nespojitý v čase i amplitudě. Tento signál vzniká vzorkováním (nespojitý v čase) a kvantováním (nespojitý v amplitudě) analogového signálu.
- Signál číslicový (digitální) – je speciálním případem nespojitého signálu, které vyjadřuje zprávu omezeným počtem hodnot.

Způsob, jakým si můžeme přizpůsobit kódovanou zprávu (signál) pro přenos konkrétním přenosovým kanálem, se nazývá modulace. Ta se provádí na straně vysílače, zatímco na straně přijímače se provádí inverzní proces demodulace). Před vysvětlením těchto procesů si dále rozdělíme signály, s kterými budeme pracovat:

- Signál modulační - vstupní analogový či digitální signál.
- Signál nosný - vysokofrekvenční harmonický signál.
- Signál modulovaný - výstupní harmonický signál.

Modulací se rozumí proces, při kterém dochází k ovlivěnění určitého parametru (změna charakteru) nosného signálu na základě vstupního modulačního signálu, který je nositelem zprávy. Změnou charakteru se myslí amplituda, frekvence a počáteční fáze nosného signálu. Výsledkem modulačního procesu je modulovaný signál na výstupu. Demodulací získáme z modulovaného signálu zpátky vstupní modulační signál. [3]

Matematické vyjádření harmonického signálu zní:

$$w(t) = A\sin(\omega t + \varphi_0) \tag{2.1.1}$$

kde:

- A je kladná konstanta, amplituda.

- t je reálná konstanta, čas.
- φ_0 je reálná konstanta, počáteční fáze, tj. fáze pro okamžik $t = 0$.
- ω je kladná konstanta, úhlový kmitočet, frekvence je na ni vázaná vztahem:

$$\omega = \frac{2\pi}{T} = 2\pi f \tag{2.1.2}$$

Z rovnic 2.1.1 a 2.1.2 [2] budeme vycházet při simulaci modulací.

2.2 Základní rozdělení

Modulace se dělí podle typu modulačního a nosného signálu [3] na:

- Analogová modulace – značí, že modulační signál nesoucí zprávu je analogový a nosný signál je harmonického průběhu. Tyto modulace se označují jako amplitudová modulace AM (amplitude modulation), frekvenční modulace FM (frequency modulation) a fázová modulace PM (phase modulation). Simulace analogových modulací není cílem této práce a nebudeme je dále rozebírat.
- Digitální modulace – značí, že modulační signál je digitální, nabývá tedy konečného počtu hodnot. Nosný signál je opět harmonického průběhu.

Digitální modulace dělíme dále na:

- Jednoduché modulace:
  - Klíčování amplitudovým zdvihem ASK (Amplitude-shift keying).
  - Klíčování frekvenčním zdvihem FSK (Frequency-shift keying).
  - Klíčování fázovým zdvihem PSK (Phase-shift keying).
- Složené modulace:
  - Kvadraturní amplitudová modulace QAM (Quadrature amplitude modulation).
    - Kombinuje amlitudové a fázové klíčování – ASK a PSK.

Při simulaci jednoduchých modulací využiji pouze dvou modulačních stavů. Použití více n modulačních stavů u složené modulace QAM se značí jako n-QAM (např. 4-QAM, 8-QAM, 16-QAM, atd.).

3 Seznámení s programem GNU Octave

3.1 Projekt GNU

GNU Octave, jak již název programu napovídá, je součástí projektu GNU, který je šířen jako svobodný software. Samotný program zahrnuje velkou škálu programů s různým využitím. Nejprve si ale uvedeme základní principy projektu a licenční podmínky, které se na program GNU Octave vztahují.

Projekt GNU byl založen v roce 1984 s cílem vytvořit svobodný UNIX-like operační systém. Samotný název GNU tvoří akronym pro „GNU's Not UNIX".

V návaznosti na tento projekt byla v roce 1985 založena Nadace pro svobodný software (Free Software Foundation) za účelem podporovat práva uživatelů počátačů používat, kopírovat, modifikovat a volně šířit počítačové programy. Dnes jsou hojně používány různé varianty operačního systému GNU s kernelem GNU/Linux nebo Nexenta OS.

Na všechny softwarové balíčky zařazené do projektu GNU (včetně programu GNU Octave) se vztahuje všeobecná veřejná licence GNU (GNU General Public Licence) označována zkratkou GNU GPL. V tomto smyslu je svobodný software takový, k němuž je k dispozici také zdrojový kód. Bývá zvykem, že jednotlivé programové balíčky mají internetové stránky, kde je možné samotný program stáhnout. Manuály (standardně v angličtině) ke stažení a diskuzní fóra, kde mohou uživatelé řešit konkrétní problémy, jsou taktéž dostupné na internetových stránkách. [5]

3.2 Vlastnosti programu

GNU Octave je vyšším programovacím jazykem primárně zaměřený na číselné výpočty. Syntaxí jazyka je velmi podobný více rozšířenému a bohatšímu programu Matlab a příkazy jsou z části kompatibilní mezi sebou.

Autorem programu je John W. Eaton [4]. Program vytvořit jako pomůcku pro studenty, aby nemuseli řešit úlohy v běžných programových jazycích jako Fortran nebo C.

GNU Octave je podobně jako Matlab založen na efektivním počítání s maticemi. Ve své základní nerozšířené verzi nemá k dispozici grafické prostředí a příkazy či předem připravené skripty se zadávají do příkazového řádku. Vkládat je možné i komplexní čísla, goniometrické, hyperbolické, statistické či logické funkce, logaritmy, cykly, podmínky, polynomy, algoritmy pro numerické řešení diferenciálních rovnic, numerickou intergraci, konverzi a rozklad obrázkových i audio dat, zpracování signálu a mnohé další.

Pro zobrazování grafických výsledků se využívá program *GNUPlot*, který umožní vykreslený obrázek následně i uložit v různých formátech (png, pdf, atd.). [4, 5]

3.3 Výhody a nevýhody GNU Octave

Volba využívaného programu je záležitostí každého uživatele. Pro oblast matematicky zaměřených programů je k dispozici velké množství programů. Ke komerčním patří např. Matlab, Maple nebo Mathematica. Svobodný software můžeme najít pod názvy jako Scilab či GNU Maxima.

S používáním programu GNU Octave jsou spojeny určité výhody a nevýhody. Kromě ceny (program je zdarma) je dobré zvážit následující klady („+“) a zápory („–“):

– K efektivnímu používání je nutné znát základy programovacího jazyka.

– Při výpočtech manipulujeme s maticemi a ty vyžadují zvláštní syntaxi.

– Program nemá grafické rozhraní a většina úkonů je prováděna v příkazovém řádku.

± GNU Octave není výukovým programem, jelikož se nezaměřuje na grafické ovládání a pohyblivé simulace. Na druhé straně se podobá programům, které mají uplatnění v reálné praxi.

+ Program byl vytvářen jako mezinárodní projekt a existuje tedy velké množství materiálů, manuálů a diskuzních portálů v řadě různých jazyků. Dále je velmi pravděpodobné, že v široké komunitě již někdo vyřešil podobný problém, na jakém pracujeme sami. Větší počet uživatelů taktéž rychleji odhalí případné nedostatky a chyby programu.

+ Programovací jazyk používaný GNU Octave má syntaxi velmi podobnou v praxi široce používanému programu Matlab. Jakmile si uživatel zvykne na základní principy a filozofii GNU Octave, případný přechod mezi programy není tak obtížný.

+ GNU Octave je multiplatformním programem. Lze ho tedy používat jak na operačním systému Windows, tak na většině distribucí Linuxu a operačním systému OS X od společnosti Apple.

+ Základní manuály můžeme v českém jazyce nalézt na internetových stránkách [7, 8].

Rozsáhlejší seznam výhod a nevýhod spojených s používáním GNU Octave můžete nalézt v literatuře věnující se práci s programem [4, 5].

3.4 Zvolení verze programu

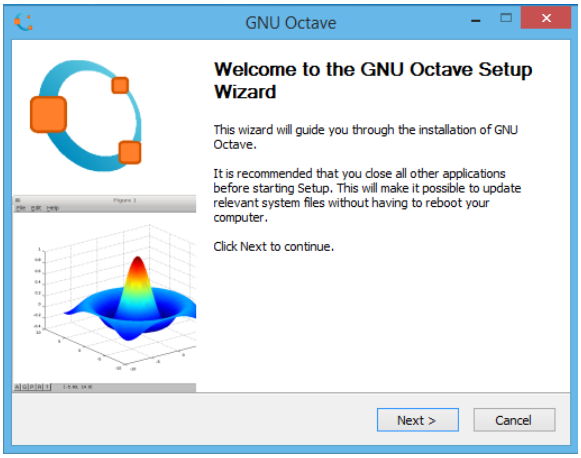
Pro simulaci modulací a demodulací jsem zvolil GNU Octave verze 3.8.1, která je volně dostupná ke stažení na domovských internetových stránkách GNU Octave [9] od 7. března 2014. S příchodem této verze bylo implementováno zkušební grafické rozhraní a očekává se, že s vydáním verze 4.0.x bude grafické rozhraní určeno jako základní.

Při práci s programem jsem používal textové uživatelské rozhraní v příkazovém řádku a v grafickém rozhraní jsem nepracoval. Podrobější seznam provedených změn na verzi 3.8.1 můžete nalézt na domovských stránkách GNU Octave v sekci novinek [10].

3.4.1 Průběh instalace

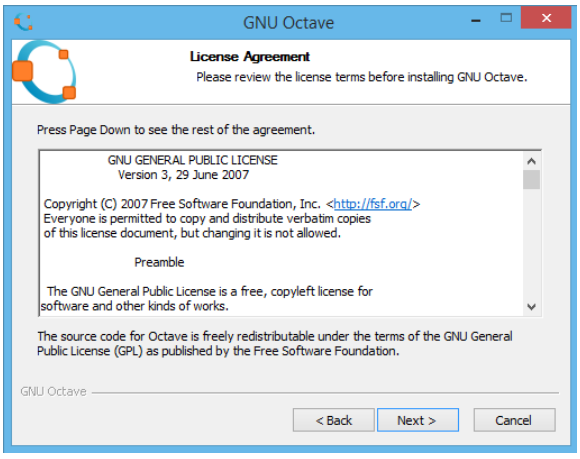
V předešlé kapitole 3.3 jsem zmínil, že GNU Octave je multiplatformní program. V rámci bakalářské práce jsem program nainstaloval pod operačním systémem Windows 7, x64. Návod, jak postupovat při instalaci na Linuxových distribucích, je popsán v literatuře pro začátečníky [4].

Instalační balíček *octave-3.8.1-installer.exe* má velikost 121 MB. Je tedy značně větší než předchozí verze a to z důvodu zabudování grafického rozhraní.



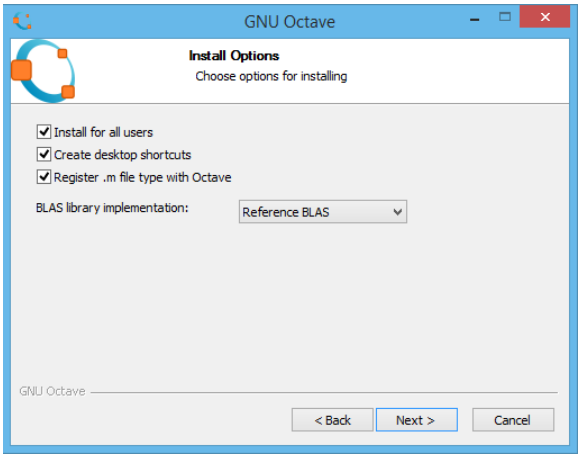
Obrázek 3.4.1: Instalace – uvítací okno

Při spuštění instalace se zobrazí uvítací okno instalátoru. Pokračujeme stisknutím tlačítka *Next*.



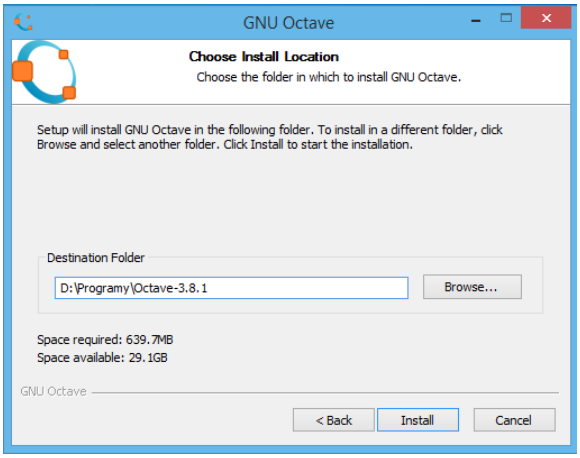
Obrázek 3.4.2: Instalace – licenční podmínky

Musíme potvrdit souhlas s licenčními podmínkami k používání programu.



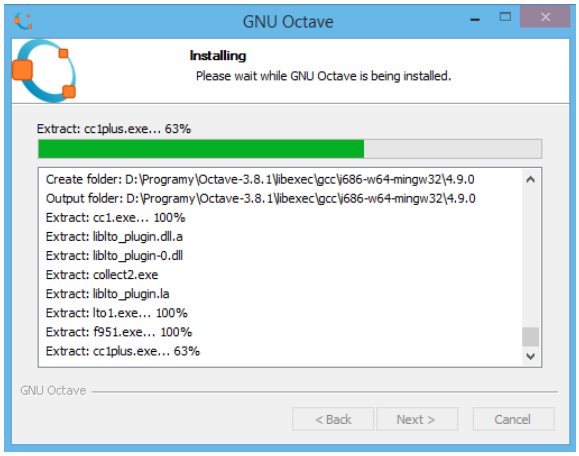
Obrázek 3.4.3: Instalace – volby instalace

V této části můžeme zvolit, zdali program bude dostupný pro všechny uživatele registrované na počítači. Potvrdíme vytvoření zástupce umístěným na ploše. Soubory přípony *.m* je vhodné neasociovat s programem Octave a tuto volbu odkliknout. Tyto soubory slouží k uložení skript, které obsahují náš programový kód a upravují se v některém z textových editorů (např. Notepad++).



Obrázek 3.4.4: Instalace – umístění cílového adresáře

Zvolíme cestu k cílovému adresáři. Můžeme vidět, že instalace GNU Octave vyžaduje minimálně 640 MB volného místa. Instalaci potvrdíme tlačítkem *Install*.



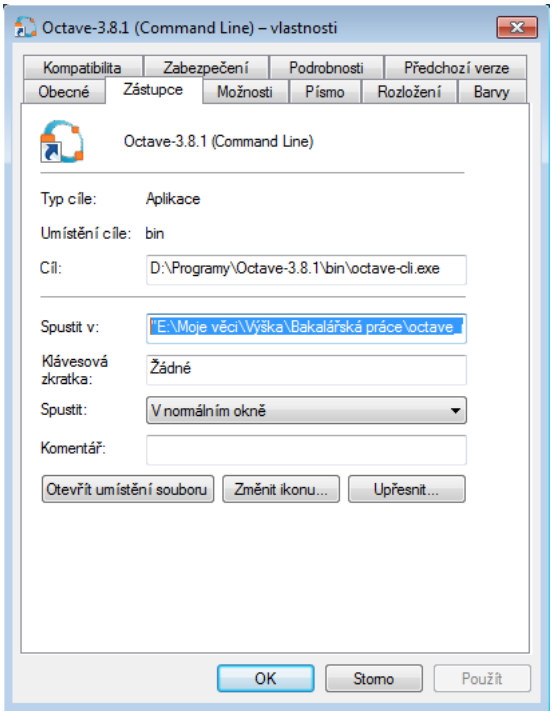
Obrázek 3.4.5: Instalace – průběh instalace

Zobrazí se průběh instalace. Po dokončení se zobrazí okno o úspěšném průběhu instalace s nabídkou spuštění programu a otevření souboru *ReadMe*. Tímto je instalace GNU Octave dokončena a můžeme program spustit.

3.4.2 První spuštění

Chceme-li pracovat na rozsáhlejším projektu, vyplatí se náš programový kód uložit do textového formátu, který můžeme libovolně ukládat, přenášet a upravovat bez ztráty dat. Takový soubor má příponu *.m* a nazývá se skriptovací soubor (zkráceně skript).

Vytvoříme si adresář, do kterého budeme skripty ukládat. Aby se GNU Octave spustil v tomto adresáři a umožnil snadné spuštění našich skriptů, klikneme pravým tlačítkem na programového zástupce *Octave-3.8.1 (Command Line)* umístěného na ploše či v nabídce start a zvolíme *Vlastnosti*.

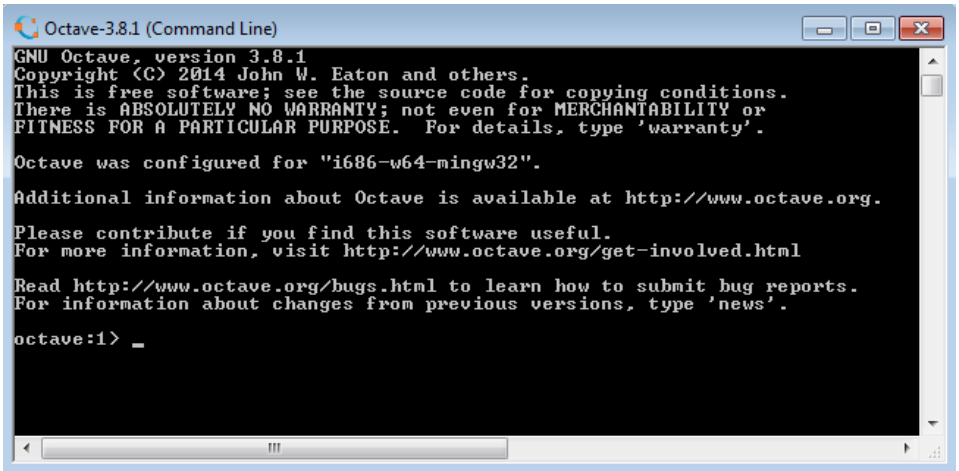


Obrázek 3.4.6: Nastavení vlastností zástupce GNU Octave

V položce *Spustit v:* nastavíme umístění vytvořeného adresáře pro ukládání skriptů. V mém případě je adresa nastavena jako:

`"E:\Moje věci\Výška\Bakalářská práce\octave_projects\".`

Změnu potvrdíme tlačítkem *OK*.



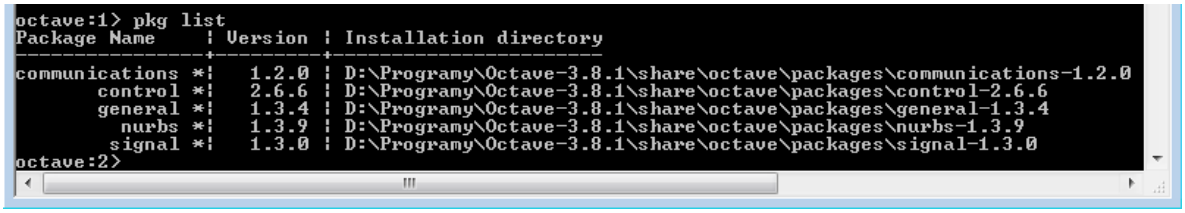
Obrázek 3.4.7: Příkazový řádek programu GNU Octave

GNU Octave nemá žádné předinstalované balíčky funkcí, ale disponuje vlastní sadou základních funkcí. Pro simulaci modulací a demodulací jsem nainstaloval tyto balíčky:

- communications 1.2.0 – Digitální komunikace, opravné kódy, modulace a galoisovo těleso.
- control 2.6.6 – Nástroje pro návrh počítačem řízeného systému.
- general 1.3.4 – Obecné nástroje pro Octave.
- nurbs 1.3.9 – Převody jednotek stupňů a radiánů.
- signal 1.3.0 – Nástroje zpracování signálu, včetně filtrovacích a vyreslovacích funkcí.

Balíčky jsou dostupné ke stažení na internetových stránkách [12]. Seznam všech funkcí dostupných instalovatelných balíčků je možné dohledat na internetové stránce [13].

Stažené balíčky vložíme do adresáře pro skripty a nainstalujeme příkazem *pkg install NÁZEV_BALÍČKU*. Příkaz *pkg load all* všechny balíčky připraví k použití. Seznam nainstalovaných balíčků můžeme zobrazit příkazem *pkg list*. [11]



Obrázek 3.4.8: Seznam nainstalovaných balíčků funkcí

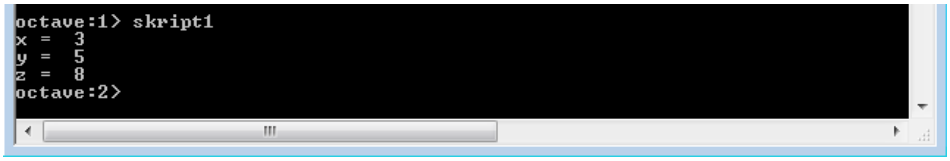
3.4.3 Spuštění našeho skriptu

Do našeho adresáře pro skriptovací soubory si vložíme dva zkušební skripty a pojmenuje je *skript1.m* a *skript2.m*.

V prvním skriptu zkusíme provést jednoduchý numerický výpočet součtu dvou proměných a zobrazit výsledek. Pro spuštění skriptu stačí zadat *NÁZEV_SOUBORU* bez přípony a skript se provede.

```
x = 3
y = 5
z = x + y
```

Výpis 1: Skriptovací soubor – skript1.m

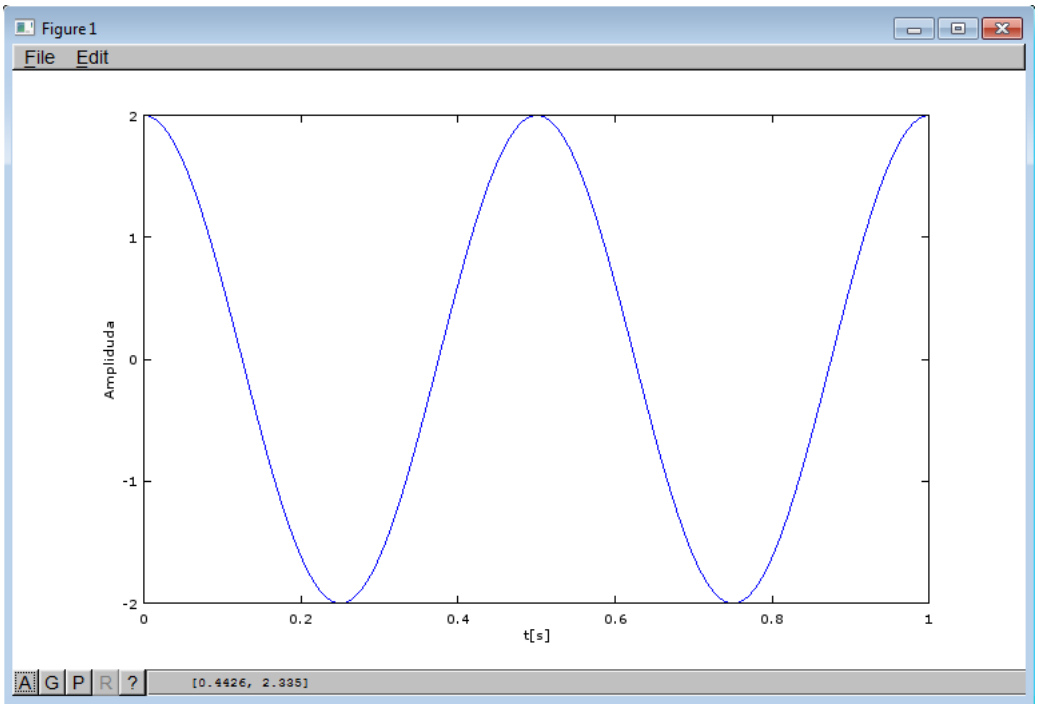


Obrázek 3.4.9: Náš první numerický výpočet

GNU Octave je schopen vykreslit i grafický výsledek a proto si zkusíme zobrazit signál harmonického průběhu. Vykreslíme jej na časovou osu o jedné vteřině. Signál je určen goniometrickou funkcí $sinus$, amplituda signálu je rovna 2, zvolená frekvence je 2 Hz, tedy dvě periody signálu za vteřinu a počáteční fáze je $\frac{\pi}{2}$.

```
t = 0:0.0001:1;
signal = 2 * sin (2*pi*2*t + pi/2);
plot(t, signal)
xlabel("t[s]")
ylabel("Ampliduda")
```

Výpis 2: Skriptovací soubor – skript2.m



Obrázek 3.4.10: Naše první vykreslení průběhu signálu

4 Simulace jednotlivých typů modulace

4.1 Modulační signál

Před začátkem tvorby skriptovacího souboru je na úvod doporučeno uvést následující dva příkazy.

```
clear all;
close all;
```

Výpis 3: Úvod skriptovacího souboru

Docílí se toho, že se při opětovném zavolání skriptu z příkazového řádku vymažou všechny hodnoty proměnných a zavřou se všechna otevřená okna.

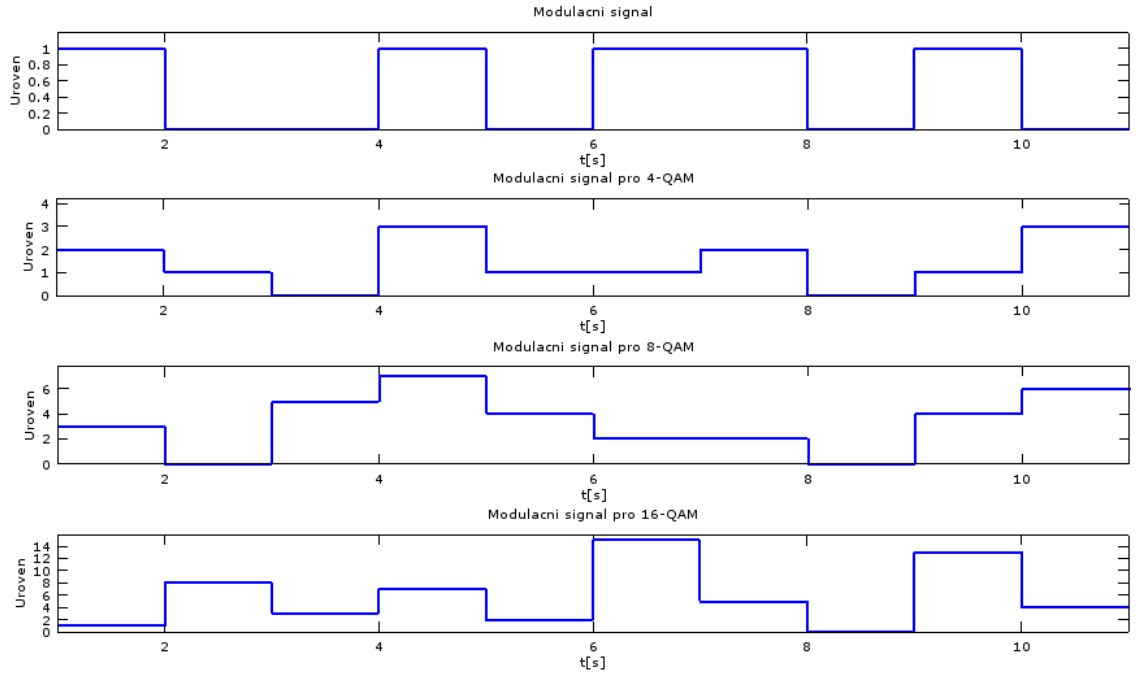
V kapitole 2.1 jsme si uvedli, že v digitálních modulacích je modulační signál taktéž digitální. Zvolíme si tedy modulační signál jako sekvenci nul a jedniček v binární soustavě a uložíme jej do jednořádkové matice, resp. pole. Takto nadefinovaný modulační signál využijeme pro jednoduché dvoustavové modulace ASK, FSK a PSK.

Pro zjednodušení vyjádření modulačního signálu o více stavech jsem použil hodnoty v desítkové soustavě, které vyjadřují zakódovanou kombinaci nul a jedniček v binární soustavě. Modulační signál o hodnotě 3 značí binární kombinaci 11, hodnota 4 značí 100 a nápodobně. Takto definované modulační signály budou využity u složených vícestavových modulací 4-QAM, 8-QAM a 16-QAM.

```
modulacni_signal = [1 0 0 1 0 1 1 0 1 0 0 1 0 0 1 1 1 0 0 1];
modulacni_signal_4QAM = [2 1 0 3 1 1 2 0 1 3 3 1 0 0 1 1 0 2 1 2];
modulacni_signal_8QAM = [3 0 5 7 4 2 2 0 4 6 6 4 5 2 7 1 3 0 2 1 4 5];
modulacni_signal_16QAM = [1 8 3 7 2 15 5 0 13 4 4 8 11 1 6 14 10 12 4];
```

Výpis 4: Použité modulační signály

Vysvětlením funkce pro vykreslovaní se budeme zabývat až v poslední kapitole 5. Pokračujeme vykreslením všech čtyř modulačních signálu.



Obrázek 4.1.1: Modulační signál o dvou, čtyřech, osmi a šesnácti stavech

4.2 Nosný signál

Nosný signál je harmonického průběhu a je vyjádřen rovnicí 2.1.1. Při procesu modulace se mění charakter nosného signálu na základě signálu modulačního. Definujme si, jak bude signál vypadat pro určité stavy modulačního signálu.

Nejprve si zvolíme základní parametry, které určují charakter nosných signálů pro ASK, FSK a PSK.

```
amp0 = 0;
amp1 = 1;
frek0  = 2;
frek1  = 6;
omega0 = 2 * pi * frek0;
omega1 = 2 * pi * frek1;
```

Výpis 5: Parametry nosného signálu ASK, FSK a PSK

Jelikož používáme modulační signál pouze o dvou stavech, parametry nosného signálu budou nabývat maximálně dvou různých hodnot.

Amplituda nosného signálu může nabývat hodnoty 0 a 1. Nosný signál je typicky vysokofrekvenční, ale pro přehlednost v rámci simulace je frekvence signálu rovna 2 a 6 Hz. Hodnoty počáteční fáze si určíme později.

Dále je nutné vymezit časovou oblast, ve které signál vykreslíme.

```
vzorku_na_stav = 1000;
pocet_intervalu = 10;
t = 0:1/vzorku_na_stav:pocet_intervalu+1;
```

Výpis 6: Definice časové oblasti

Pro snadnější manipulaci s parametry simulace je časová oblast definována s pomocnými proměnnými.

S navyšováním počtu vzorků na stav je vykreslený signál více věrohodný, ale numerické výpočty trvají delší čas. Zvolil jsem 1000 vzorků nosného signálu na jeden modulační stav, ale pro zrychlení simulace na slabších počítačích doporučuji hodnotu vzorků snížit na 100.

Počet intervalů udává, že bude vykresleno celkem 10 různých hodnot modulačního signálu na časové ose (a jedna hodnota navíc, která již nebude zobrazena). Přiřadíme určené parametry k nosným signálům pro ASK, FSK a PSK pro určité hodnoty modulačních stavů:

```
s_ASK_0 = amp0 * sin(omega0 * t);
s_ASK_1 = amp1 * sin(omega0 * t);
s_FSK_0 = amp1 * sin(omega0 * t);
s_FSK_1 = amp1 * sin(omega1 * t);
s_PSK_0 = amp1 * sin(omega0 * t + (0*pi));
s_PSK_1 = amp1 * sin(omega0 * t + (1*pi));
```

Výpis 7: Nosný signál ASK, FSK a PSK v konkrétních modulačních stavech

Jelikož se jedná o jednoduché modulace, pro každý typ modulace se mění pouze jeden z parametrů. Pro nosný signál ASK měníme amplitudu signálu na hodnotu 0 a 1, pro FSK se mění frekvence nosného signálu mezi 2 a 6 Hz a u nosného signálu typu PSK měníme hodnotu počáteční fáze mezi 0 a $1 * \pi$.

4.3 Modulace

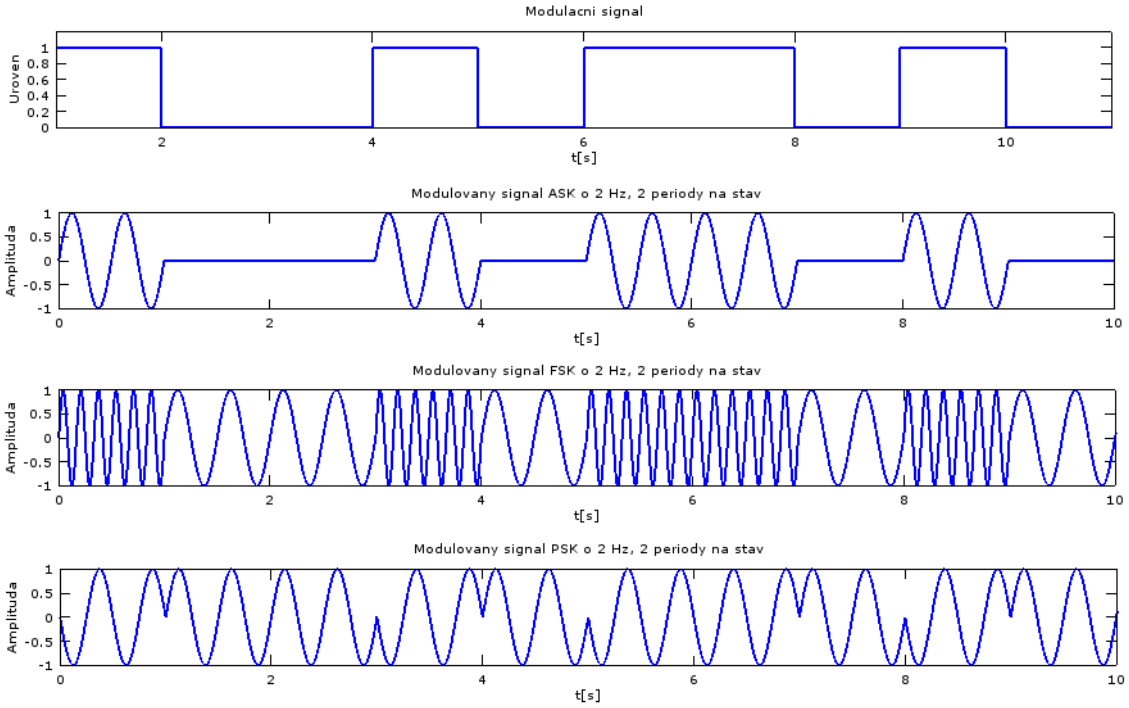
4.3.1 ASK, FSK a PSK

Tradičním způsobem, jak manipulovat s hodnotami uloženými v datovém typu pole, je využití cyklů, podmínek a pomocné proměnné zvané index.

Modulace typu ASK, FSK a PSK vypadá následovně.

```
for(i = 1 : pocet_intervalu+1)
  for(j = (i-1) * vzorku_na_stav : i * vzorku_na_stav)
    if(modulacni_signal(i) == 0)
      modulovany_signal_ASK(j+1) = s_ASK_0(j+1);
      modulovany_signal_FSK(j+1) = s_FSK_0(j+1);
      modulovany_signal_PSK(j+1) = s_PSK_0(j+1);
    elseif(modulacni_signal(i) == 1)
      modulovany_signal_ASK(j+1) = s_ASK_1(j+1);
      modulovany_signal_FSK(j+1) = s_FSK_1(j+1);
      modulovany_signal_PSK(j+1) = s_PSK_1(j+1);
    endif
  endfor
endfor
```

Výpis 8: Modulace ASK, FSK a PSK



Obrázek 4.3.1: Modulované signály ASK, FSK a PSK

V GNU Octave se syntaxe cyklu *for* a podmínky *if* velmi podobá programovacímu jazyku C. Každý jednotlivý cyklus a podmínka musí být zakončena příkazem *endfor* či *endif*, aby se správně rozpoznal jejich rozsah.

Modulace ASK, FSK a PSK se provádí ve dvou vnořených cyklech, kdy se vnější provede podle počtu určených intervalů. Vnitřní cyklus se provádí na základě počtu vzorků

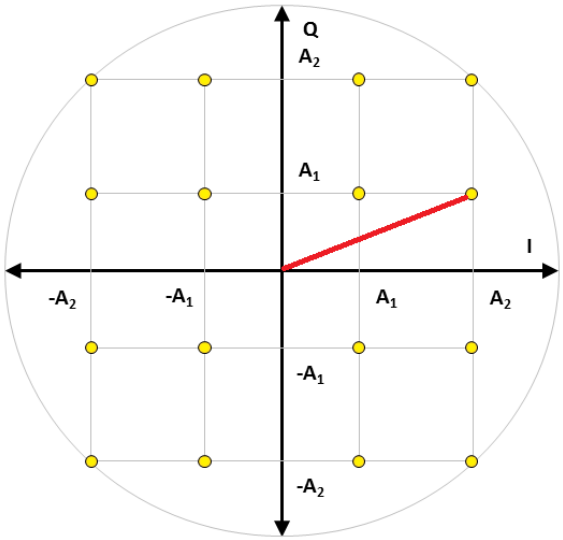
nosného signálu. Uvnitř vnitřního cyklu vznikají konečné modulované signály ASK, FSK a PSK.

Hodnoty modulovaného signálu se určují podmíněným rozhodnutím na základě hodnoty modulačního signálu. Když se v daném intervalu nachází modulační stav v hodnotě 0, jsou do modulovaného signálu dosazeny hodnoty, které odpovídají hodnotám nosného signálu pro patřičnou hodnotu modulačního signálu.

Počet hodnot výsledného modulovaného signálu je dán součinem počtu intervalů modulačního signálu a počtu vzorků nosného signálu. Výsledek tedy činí celkem 11 001 hodnot (1 000 hodnot je navíc, které ale nejsou vykresleny a 1 hodnota je pro $t = 0$).

4.3.2 Konstelační diagram

Před začátkem simulace složených vícestavových modulací si vysvětlíme pojem konstelační diagram. Konstalační diagramy se vztahují k jednotlivým typům modulací, jako například 16-QAM.



Obrázek 4.3.2: Ideální konstelační diagram pro 16-QAM [3]

Modulační stavy se dají interpretovat pomocí konstelačního diagramu, který tvoří komplexní rovinu, kde reálná osa odpovídá ose I (v angl. In Phase) a imaginární osa odpovídá ose Q (v angl. Quadrature). [3]

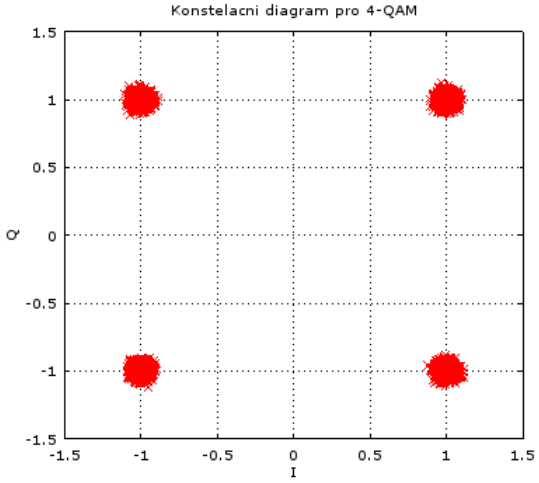
Do obrázku 4.3.2 jsem dokreslil pomyslný fázor k jednomu z modulačních stavů. Délka fázoru od středu grafu k modulačnímu stavu určuje, jaká bude hodnota amplitudy nosného signálu pro daný modulační stav. Dále pak úhel, který svírá fázor s reálnou osou I, určuje hodnotu počáteční fáze nosného signálu pro daný modulační stav.

4.3.3 4-QAM

Než začneme s modulací, nejprve si vykreslíme konstelační diagram pro 4-QAM. Pro lepší zřetelnost vykreslených modulačních stavů je přidán šum. Vysvětlení využití šumu bude blíže popsáno v kapitole 4.4.

```
pocet_stavu = 4;
pocet_prvku_IQ_4QAM = 10000;
d_4QAM = randint(1, pocet_prvku_IQ_4QAM, pocet_stavu);
c_4QAM = [1+1j, −1+1j, −1−1j, 1−1j];
y_4QAM = genqammod(d_4QAM, c_4QAM);
```

Výpis 9: Určení modulačních stavů 4-QAM



Obrázek 4.3.3: Konstelační diagram 4-QAM s nízkou hodnotou šumu

Určíme si čtyři modulační stavy a deset tisíc prvků, které se vykreslí do konstelačního diagramu. Funkce *randint* [6, 13] vytvoří matici o jednom řádku a deseti tisíci sloupcích (resp. pole, které má 10 000 hodnot) s rozmezím hodnot 0–4.

Hodnoty v poli *c_4QAM* určují rozložení modulačních stavů v komplexní rovině. Funkce *genqammod* [6, 13] slouží k namodulování sekvence 10 000 hodnot na množinu komplexních souřadnic specifikující rozdělení modulačních stavů v konstelačním diagramu.

Chceme-li modulovat pro 4-QAM dle námi zvoleného kostelačního diagramu, musíme nejprve vypočítat hodnotu amplitudy a počáteční fáze pro každý modulovaný stav.

Modulační stav v komplexní rovině je zapsán ve tvaru komplexně sdružého čísla:

$$z = x + iy \tag{4.3.1}$$

kde:

- x a y jsou reálná čísla.

Hodnotu amplitudy nosného signálu vypočítáme z absolutní hodnoty komplexně sdruženého čísla. Ta je dána rovnicí:

$$|z| = \sqrt{x^2 + y^2} \tag{4.3.2}$$

Hodnota počáteční fáze φ_0 je rovna tzv. argumentu komplexně sdruženého čísla. Argument vypočteme jednou z dvou rovnic za základě hodnoty x.

Když je $x > 0$, tak platí:

$$arg(z) = \varphi_0 = \tan^{-1}\left(\frac{y}{x}\right) \tag{4.3.3}$$

Když je $x < 0$, tak platí:

$$arg(z) = \varphi_0 = \tan^{-1}\left(\frac{y}{x}\right) \pm \pi \tag{4.3.4}$$

Vypočteme konkrétné hodnoty amplitudy a počáteční fáze nosného signálu z rovnic 4.3.2, 4.3.3 a 4.3.4. [14]

Rešení pro jeden z modulačních stavů $z_0 = 1 + 1j$:

$$\begin{aligned} |z_0| &= \sqrt{x^2 + y^2} \\ &= \sqrt{1^2 + 1^2} \\ &= \sqrt{2} \end{aligned}$$

$$\begin{aligned} arg(z_0) = \varphi_0 &= \tan^{-1}\left(\frac{y}{x}\right) \\ &= \tan^{-1}\left(\frac{1}{1}\right) \\ &= \tan^{-1}(1) \\ &= \frac{\pi}{4}\text{ , respektive } 45^\circ \end{aligned}$$

Pro modulační stav o hodnotě $1 + 1j$ je amplituda nosného signálu rovna $\sqrt{2}$ a hodnota počáteční fáze je rovna $\frac{\pi}{4}$. Výpočet bychom opakovali pro zbylé tři modulační stavy.

Je zjevné, že amplituda je pro všechny stavy modulace 4-QAM shodná. Určíme tedy parametry všech nosných signálů, které odpovídají daným modulačním stavům.

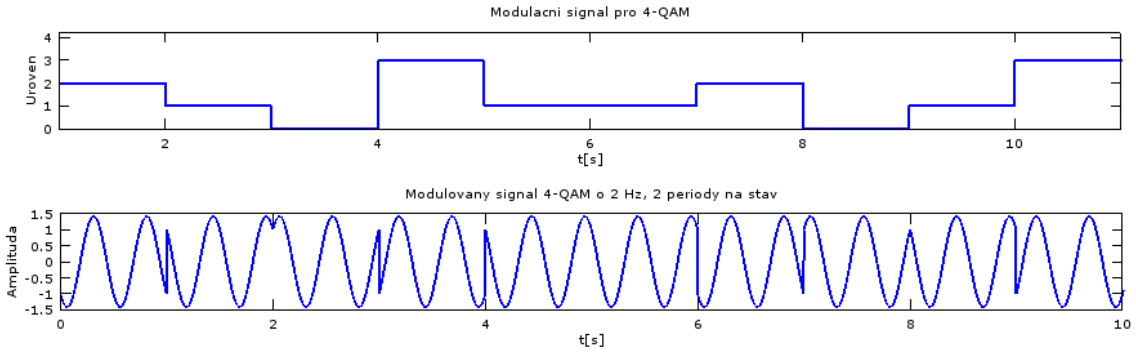
```
amp2 = sqrt(2);
s_4QAM_0 = amp2 * sin(omega0 * t + (1/4*pi));
s_4QAM_1 = amp2 * sin(omega0 * t + (3/4*pi));
s_4QAM_2 = amp2 * sin(omega0 * t + (5/4*pi));
s_4QAM_3 = amp2 * sin(omega0 * t + (7/4*pi));
```

Výpis 10: Parametry nosných signálů 4-QAM

Konečná modulace se provádí obdobným způsobem jako u modulace typu ASK, FSK a PSK u výpisu 8.

```
for(i = 1 : pocet_intervalu+1)
  for(j = (i-1) * vzorku_na_stav : i * vzorku_na_stav)
    if (modulacni_signal_4QAM(i) == 0)
      modulovany_signal_4QAM(j+1) = s_4QAM_0(j+1);
    elseif (modulacni_signal_4QAM(i) == 1)
      modulovany_signal_4QAM(j+1) = s_4QAM_1(j+1);
    elseif (modulacni_signal_4QAM(i) == 2)
      modulovany_signal_4QAM(j+1) = s_4QAM_2(j+1);
    elseif (modulacni_signal_4QAM(i) == 3)
      modulovany_signal_4QAM(j+1) = s_4QAM_3(j+1);
    endif
  endfor
endfor
```

Výpis 11: Modulace 4-QAM



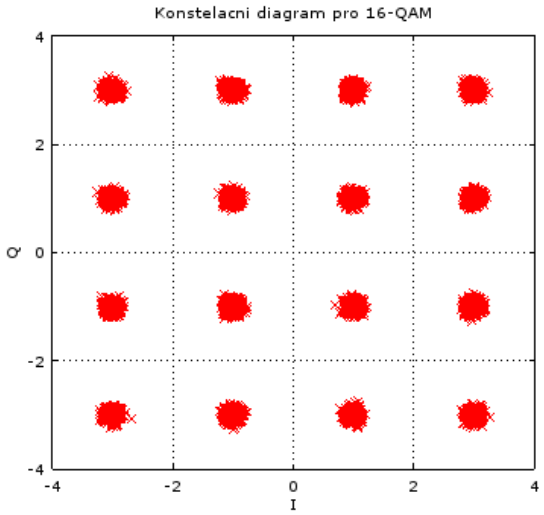
Obrázek 4.3.4: Modulace 4-QAM

4.3.4 16-QAM

Jednotlivé modulační stavy si opět nadefinujeme a zobrazíme do konstelačního diagramu. Symbol ↪ značí pokračování předchozího řádku a není součástí kódu.

```
c_16QAM = [-3-3j, -3-1j, -3+3j, -3+1j, -1-3j, -1-1j, -1+3j, -1+1j, 3-3j, 3-1j, 3+3j, 3+1j,
    ↪ 1-3j, 1-1j, 1+3j, 1+1j];
```

Výpis 12: Určení modulačních stavů 16-QAM



Obrázek 4.3.5: Konstelační diagram 16-QAM s nízkou hodnotou šumu

S narůstajícím počtem modulačních stavů je příliš pracné a neefektivní využívat modulačního algoritmu, který vyžaduje úpravu kódu pro každý odlišný typ modulace.

Navrhl jsem tedy více automatizovaný modulační algoritmus pro modulaci 16-QAM. Jednoduchou úpravou názvosloví proměnných je možné tento algoritmus použít pro libovolné složené vícestavové modulace.

```
for(i = 1 : pocet_intervalu+1)
  s_16QAM_n = abs(c_16QAM(modulacni_signal_16QAM(i)+1)) * sin(omega0 * t + (angle(
      ↪ c_16QAM(modulacni_signal_16QAM(i)+1))));
  for(j = (i-1) * vzorku_na_stav : i * vzorku_na_stav)
    modulovany_signal_16QAM(j+1) = s_16QAM_n(j+1);
  endfor
endfor
```

Výpis 13: Modulace 16-QAM

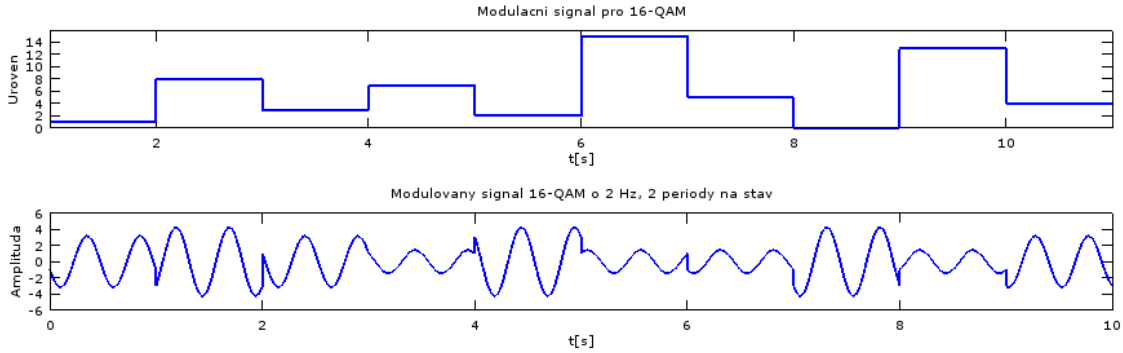
Modulace se opět provádí ve dvou vnořených cyklech, ale nyní jsou nosnému signálu dosazeny parametry na základě námi určených modulačních stavů (*c_16QAM*), které jsou voleny dle hodnoty vstupního modulačního signálu.

Modulační stavy v poli hodnot *c_16QAM* svým pořadím odpovídají hodnotě úrovně vstupního modulačního signálu. Modulační stav o hodnotě $-3-3j$ odpovídá vstupnímu modulačnímu signálu úrovně 0, modulační stav o hodnotě $-3-1j$ poté odpovídá úrovni 1 vstupního modulačního signálu a tak dál.

Chceme-li určit parametry amplitudy a počáteční fáze nosného signálu pro daný modulační stav, vypočítáme absolutní hodnotu a argument komplexně sdruženého čísla z rovnic 4.3.2 a 4.3.3.

Mezi vlastní sadu základních funkcí programu GNU Octave patří i funkce pro komplexní aritmetiku [11]. K výpočtu absolutní hodnoty komplexně sdruženého čísla použijeme funkce *abs* [4, 11]. Pro získání argumentu komplexně sdruženého čísla je možné použít jednu z identických funkcí *arg* či *angle* [11].

Tímto způsobem se dá vytvořit modulovaný signál libovolného typu složené vícestavové modulace. Pro modulace, které mění i frekvenci nosného signálu, je nutné zavést další pole o počtu hodnot frekvencí shodující se s počtem modulačních stavů. S tímto polem by se pracovalo stejně jako s polem *c_16QAM*.



Obrázek 4.3.6: Modulace 16-QAM

4.4 Přidání šumu

Šumem se myslí vedlejší informace, která nějakým způsobem mění signál. Šum není možné kompletně odstranit, ale může být minimalizován. Šum je vždy náhodný a nejasný.

Obecně při průchodu užitečného signálu komunikačních kanálem platí, že se šum k signálu přičítá (tzv. aditivní šum). Tímto se zhoršuje kvalita vysílaného signálu, který se následně přijímá na straně přijímače.

Šum v kanálu může mít původ z přírodních rušivých zdrojů, například blesk, bouřka či atmosferické změny počasí. Dále může být generován umělými zdroji, jako jsou vedení vysokého napětí, jiskření motorů nebo elektromagnetické rušení z přístrojů pracující s měničem proudu. [1]

V teoretických úvahách se předpokládá, že šum působící na signál v komunikačním kanálu je aditivní bílý Gaussovský šum AWGN (Additive White Gaussian Noise). Za aditivní je považován šum, který se lineárně sčítá s užitečným signálem, aniž by docházelo k vzájemným intermodulacím. Pojem bílý je označení šumu, jehož výkonová spektrální hustota není závislá na frekvenci. Gaussovský vyjadřuje, že rozložení okamžitých amplitud se řídí Gaussovskou distribucí. [15]

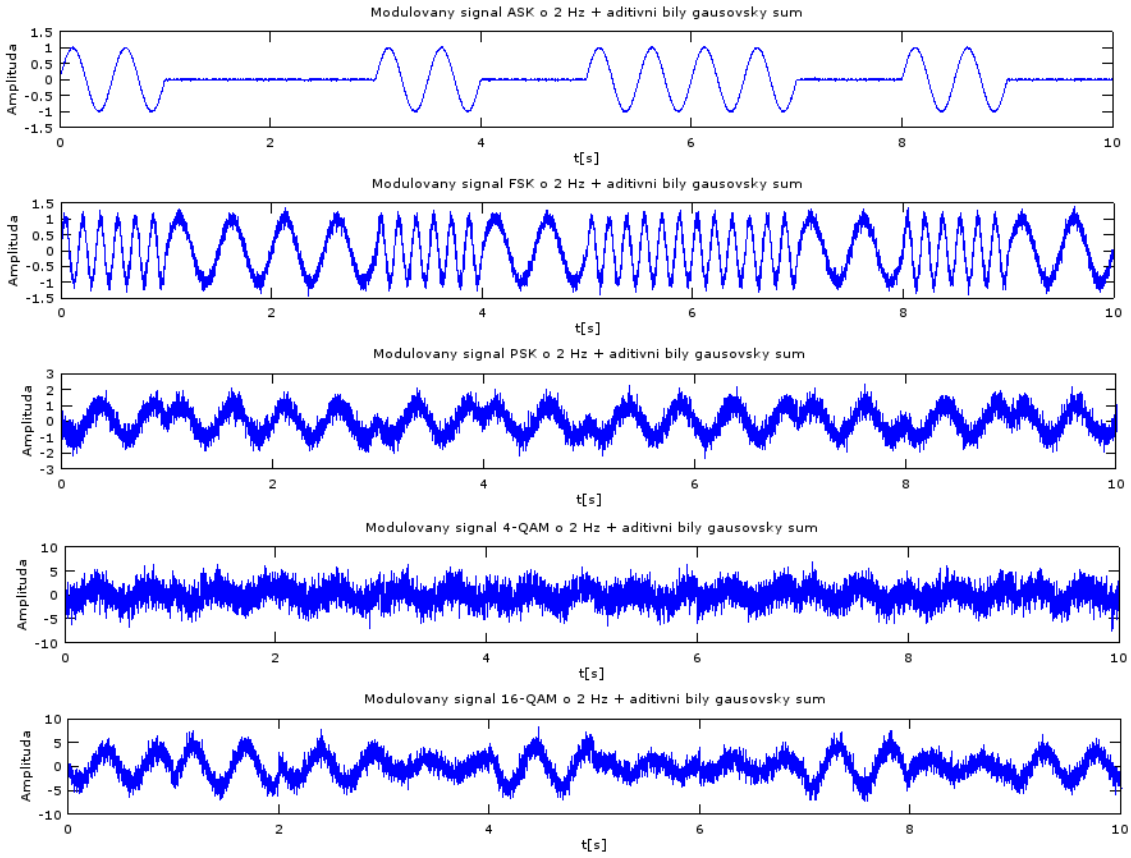
Důležitý uváděný parametr je poměr výkonu signálu k výkonu šumu SNR (Signal-to-noise ratio). Jedná se o bezrozměrnou jednotku, která se standardně udává v dB (anglicky i česky Decibel) a vyjadřuje, kolikrát je úroveň výkonu užitečného signálu vyšší oproti

úrovni výkonu nežádoucího šumu. S klesající hodnotou SNR následně roste zhoršení kvality užitečného signálu.

Než budeme modulované signály zpátky demodulovat, přičteme k nim šum o různých hodnotách SNR.

```
modulovany_signal_ASK_S = awgn(modulovany_signal_ASK, 30, 'measured');
modulovany_signal_FSK_S = awgn(modulovany_signal_FSK, 15, 'measured');
modulovany_signal_PSK_S = awgn(modulovany_signal_PSK, 5, 'measured');
modulovany_signal_4QAM_S = awgn(modulovany_signal_4QAM, −5, 'measured');
modulovany_signal_16QAM_S = awgn(modulovany_signal_16QAM, 5, 'measured');
```

Výpis 14: Přidání šumu k modulovanýcm signálům



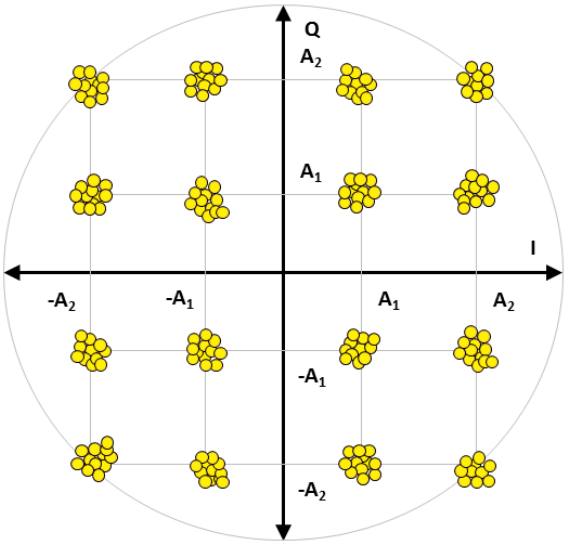
Obrázek 4.4.1: Modulované signály s přidaným šumem

Funkce pro přičtení AWGN k harmonickému signálu se nazývá *awgn* [6, 13]. Prvním argumentem funkce je cílený harmonický signál. Druhý argument označuje hodnotu SNR. Zvolil jsem pro každý modulovaný signál různou úroveň SNR, abych znázornil, jak

se mění kvalita modulovaného signálu s měnícím se SNR. V základu se považuje výkon původního modulovaného signálu za nulový a argument funkce *measured* určí, že před přičtením šumu je výkon původního signálu nejprve změřen.

S rostoucím počtem modulačních stavů se u modulací více projevuje náchylnost na rušení, které je přítomné v každém přenosovém kanálu. Pro bezchybné přenosy se u vícestavových modulací vyžaduje vyšší odstup signálu od šumu SNR.

V zobrazeném konstelačním diagramu na obrázku 4.3.2 jsou polohy jednotlivých modulačních signálu ideální, tedy vždy stejné. V případě zhoršení podmínek přenosu z důvodu zvyšování úrovně šumu (a tedy snižování hodnoty SNR), začne docházet ke kolísání modulačních stavů kolem jejich ideální polohy. Při vysokém rozptylu vychylujících se modulačních hodnot může dojít ke kolizi a tedy k chybnému vyhodnocení modulačního stavu.

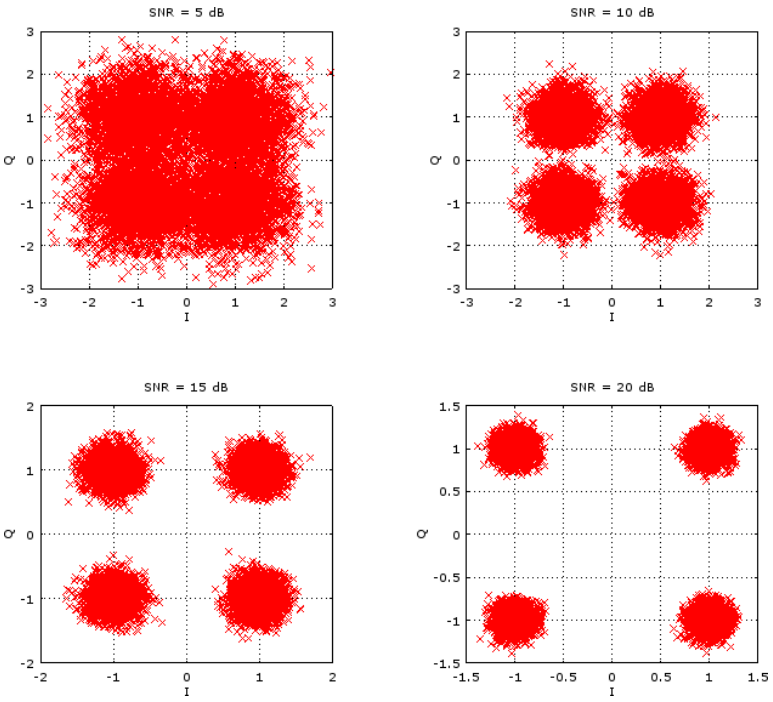


Obrázek 4.4.2: Konstelační diagram pro 16-QAM při nižším poměru SNR [3]

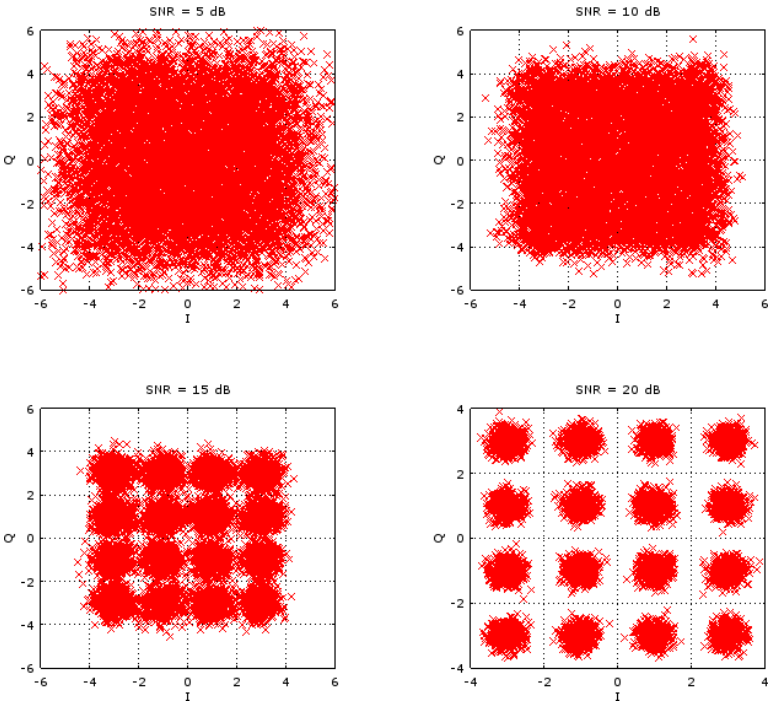
K našim vykresleným konstelačním diagramům modulace 4-QAM a 16-QAM přidáme AWGN pro ověření různé náchylnosti na rušení s měnícím se počtem modulačních stavů. Hodnoty SNR jsou dány v rozsahu 5-20 dB.

```
IQ_4QAM_1 = awgn(y_4QAM, 5, 'measured');
IQ_4QAM_2 = awgn(y_4QAM, 10, 'measured');
IQ_4QAM_3 = awgn(y_4QAM, 15, 'measured');
IQ_4QAM_4 = awgn(y_4QAM, 20, 'measured');
IQ_16QAM_1 = awgn(y_16QAM, 5, 'measured');
IQ_16QAM_2 = awgn(y_16QAM, 10, 'measured');
IQ_16QAM_3 = awgn(y_16QAM, 15, 'measured');
IQ_16QAM_4 = awgn(y_16QAM, 20, 'measured');
```

Výpis 15: Přidání šumu do konstelačních diagramů 4-QAM, 16-QAM



Obrázek 4.4.3: Konstelační diagramy 4-QAM s přidaným šumem



Obrázek 4.4.4: Konstelační diagramy 16-QAM s přidaným šumem

4.5 Demodulace

4.5.1 ASK, FKS a PSK

V průběhu demodulace převádíme výstupní modulovaný signál zpátky na vstupní modulační signál a pro modulace typu ASK, FSK a PSK ji provedeme následovně.

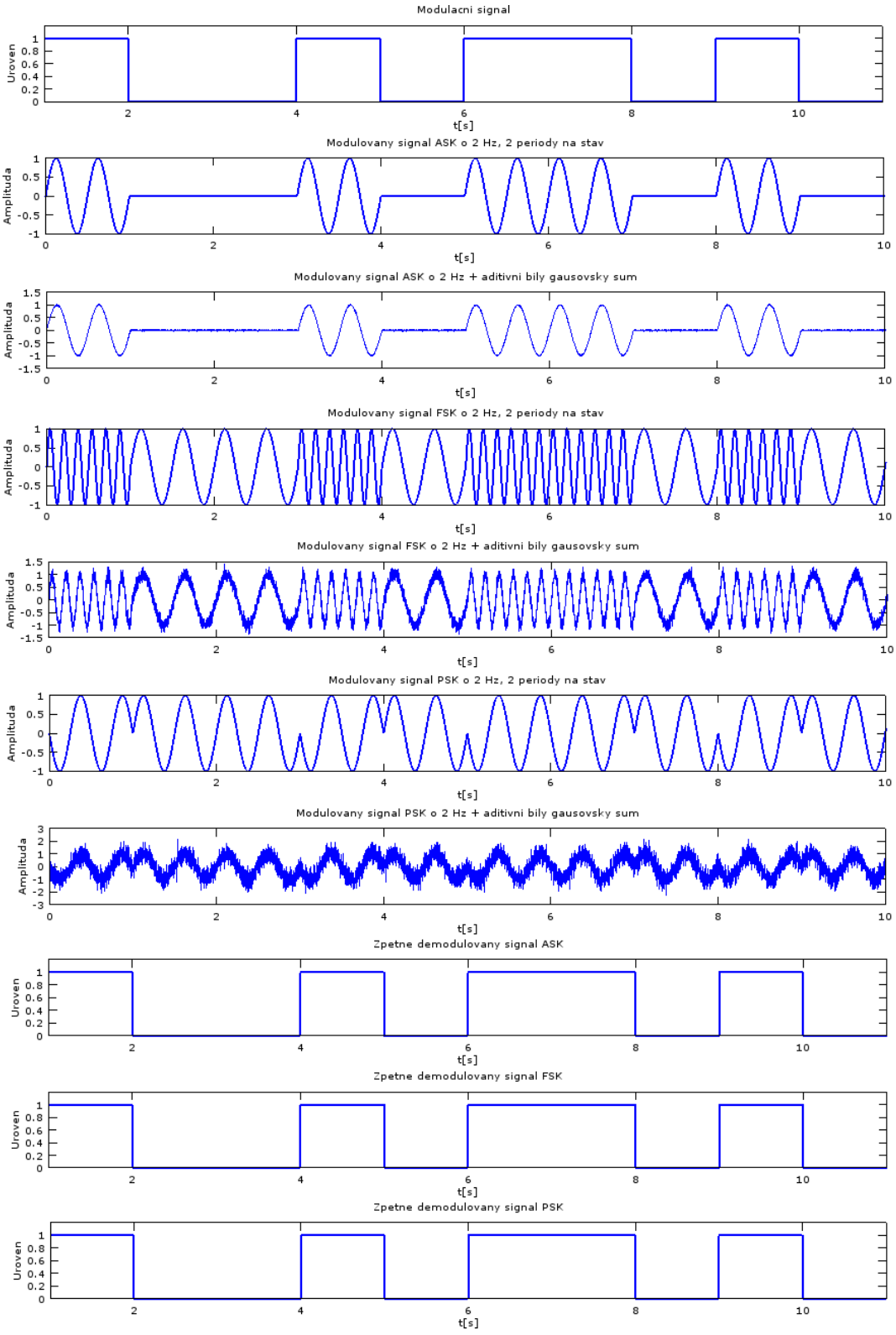
```
for(i = 1 : pocet_intervalu+1)
  demodulovany_signal_ASK_S(i) = 0;
  demodulovany_signal_FSK_S(i) = 0;
  demodulovany_signal_PSK_S(i) = 0;
  for(j = 0 : vzorku_na_stav)
    modulovany_ve_stavu_ASK(j+1) = modulovany_signal_ASK_S((i−1)∗vzorku_na_stav+j+1);
    modulovany_ve_stavu_FSK(j+1) = modulovany_signal_FSK_S((i−1)∗vzorku_na_stav+j+1);
    modulovany_ve_stavu_PSK(j+1) = modulovany_signal_PSK_S((i−1)∗vzorku_na_stav+j+1);
    referencni_nosna_ASK(j+1) = s_ASK_1((i−1)∗vzorku_na_stav+j+1);
    referencni_nosna_FSK(j+1) = s_FSK_1((i−1)∗vzorku_na_stav+j+1);
    referencni_nosna_PSK(j+1) = s_PSK_1((i−1)∗vzorku_na_stav+j+1);
  endfor
  if(corr(modulovany_ve_stavu_ASK, referencni_nosna_ASK) > 0.2)
    demodulovany_signal_ASK_S(i) = 1;
  endif
  if(corr(modulovany_ve_stavu_FSK, referencni_nosna_FSK) > 0.2)
    demodulovany_signal_FSK_S(i) = 1;
  endif
  if(corr(modulovany_ve_stavu_PSK, referencni_nosna_PSK) > 0)
    demodulovany_signal_PSK_S(i) = 1;
  endif
endfor
```

Výpis 16: Demodulace ASK, FSK a PSK

Abychom mohli signál demodulovat, budeme vždy různě porovnávat dva signály. Prvním signálem je skutečný modulovaný signál pro daný stav a druhým je tzv. ideální referenční nosný signál. Počet referenčních nosných signálu je dán počtem modulačních stavů a jejich charakter jsme si definovali před začátkem modulace. U jednoduchých modulací typu ASK, FSK a PSK jsme použili vstupní modulační signál o dvou stavech. Referenční nosné signály jsou tedy dva, ale na základě vylučovací metody je dostačující použít v kódu demodulace pouze jeden.

Aby měla demodulace vypovídající hodnotu o náchylnosti na rušení, použijeme modulované signály s přidaným AWGN z kapitoly 4.4.

K porovnání ideálního a skutečného signálu použijeme tzv. korelace [2], která se provádí funkcí *corr* [11]. Korelace značí vzájemný vztah mezi dvěma veličinami. Když se jedna z nich mění, ve vztahu k ní se mění i druhá veličina a naopak. Míru korelace vyjadřuje korelační koeficient, který nabývá hodnoty od –1 do +1. Hodnota +1 značí zcela přímou závislost a hodnota –1 poté udává zcela nepřímou závislost. Pokud je korelační koeficient roven 0, pak mezi veličinami není zjistitelná lineární závislost. Rozhodneme o úrovni demodulovaného signálu na základě hodnoty korelačního koeficientu.



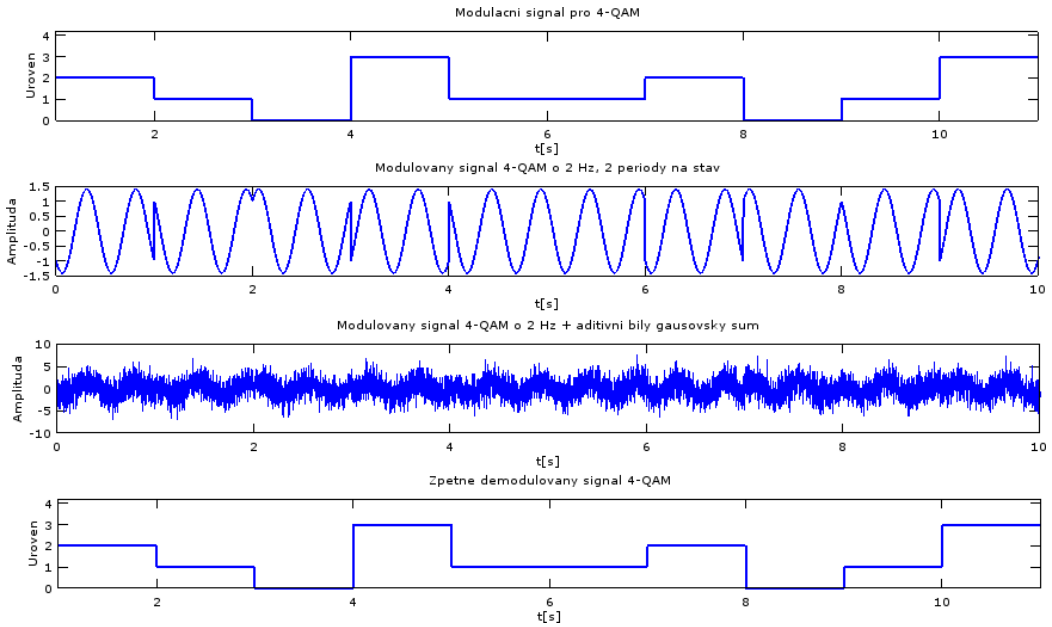
Obrázek 4.5.1: Celková modulace a demodulace ASK, FSK a PSK

4.5.2 4-QAM

Demodulace se provádí totožně jako u výpisu 16 pro ASK, FSK a PSK, ale nyní použijeme čtyři referenční nosné signály z důvodu čtyř modulačních stavů.

```
for(i = 1 : pocet_intervalu+1)
  for(j = 0 : vzorku_na_stav)
    modulovany_ve_stavu_4QAM(j+1) = modulovany_signal_4QAM_S((i-1)*vzorku_na_stav+j+1);
    referencni_nosna_4QAM_0(j+1) = s_4QAM_0((i-1)*vzorku_na_stav+j+1);
    referencni_nosna_4QAM_1(j+1) = s_4QAM_1((i-1)*vzorku_na_stav+j+1);
    referencni_nosna_4QAM_2(j+1) = s_4QAM_2((i-1)*vzorku_na_stav+j+1);
    referencni_nosna_4QAM_3(j+1) = s_4QAM_3((i-1)*vzorku_na_stav+j+1);
  endfor
  if(corr(modulovany_ve_stavu_4QAM, referencni_nosna_4QAM_0) > 0.2)
    demodulovany_signal_4QAM_S(i) = 0;
  elseif(corr(modulovany_ve_stavu_4QAM, referencni_nosna_4QAM_1) > 0.2)
    demodulovany_signal_4QAM_S(i) = 1;
  elseif(corr(modulovany_ve_stavu_4QAM, referencni_nosna_4QAM_2) > 0.2)
    demodulovany_signal_4QAM_S(i) = 2;
  elseif(corr(modulovany_ve_stavu_4QAM, referencni_nosna_4QAM_3) > 0.2)
    demodulovany_signal_4QAM_S(i) = 3;
  else
    demodulovany_signal_4QAM_S(i) = 0;
  endif
endfor
```

Výpis 17: Demodulace 4-QAM



Obrázek 4.5.2: Celková modulace a demodulace 4-QAM

4.5.3 16-QAM

Když chceme demodulovat signály o více stavech, demodulační algoritmus musíme zautomatizovat.

```
for(i = 1 : pocet_intervalu+1)
  demodulovany_signal_16QAM_S(i) = 0;
  nejvyssi_korelace = 0;
  for(p = 1 : pocet_stavu_16QAM)
    s_16QAM_n = abs(c_16QAM(p)) * sin(omega0 * t + (arg(c_16QAM(p))));
    for(j = 0 : vzorku_na_stav)
      modulovany_ve_stavu_16QAM(j+1) = modulovany_signal_16QAM_S((i-1)*vzorku_na_stav+j
          ↪ +1);
      referencni_nosna_16QAM(j+1) = s_16QAM_n((i-1)*vzorku_na_stav+j+1);
    endfor
    if (sum(abs(modulovany_ve_stavu_16QAM)) / sum(abs(referencni_nosna_16QAM)) > 0.9)
      if (sum(abs(modulovany_ve_stavu_16QAM)) / sum(abs(referencni_nosna_16QAM)) < 1.5)
        if (corr(modulovany_ve_stavu_16QAM, referencni_nosna_16QAM) >= nejvyssi_korelace)
          demodulovany_signal_16QAM_S(i) = p-1;
          nejvyssi_korelace = corr(modulovany_ve_stavu_16QAM, referencni_nosna_16QAM);
        endif
      endif
    endif
  endfor
endfor
```

Výpis 18: Demodulace 16-QAM

Kdybychom porovnávali ideální a skutečný modulovaný signál pouze na základě korelace, nebyla by demodulace pro naši modulaci typu 16-QAM možná. Z našeho navrženého konstelačního diagramu na obrázku 4.3.5 je vidět, že pro některé z modulačních stavů existují nosné signály o shodných hodnotách buď amplitudy nebo počáteční fáze. U námi definovaných jednoduchých modulací ASK, FSK a PSK a složené 4-QAM k tomuto problému dojít nemůže.

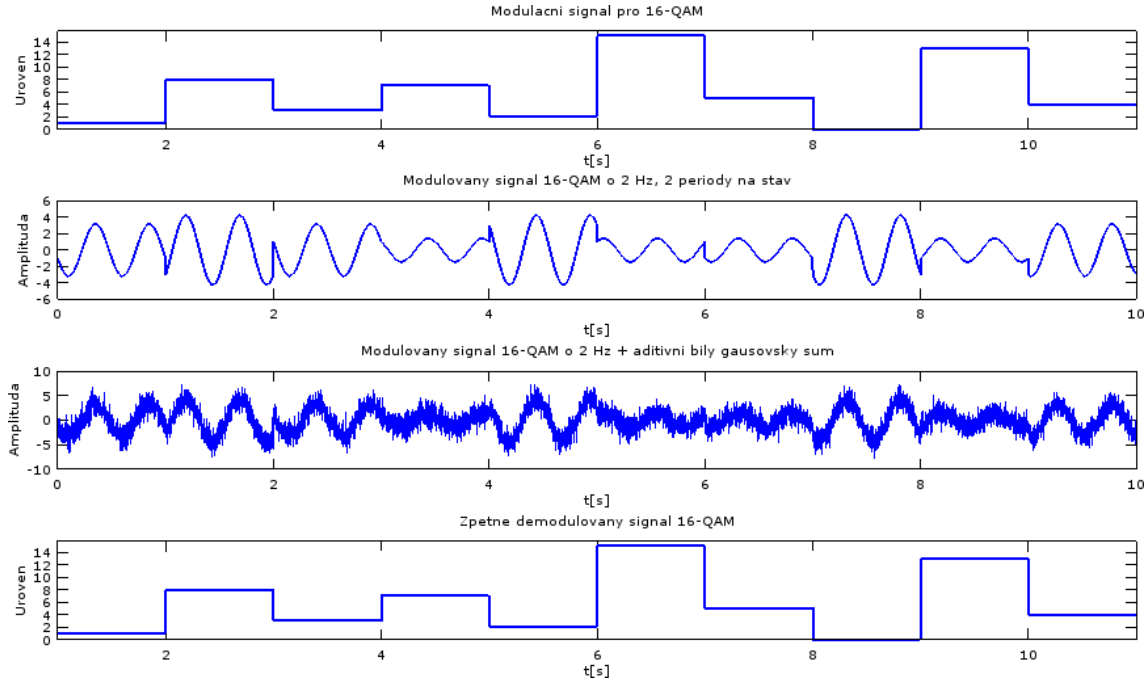
Korelací dvou ideálních signálů s totožnou počáteční fází a různou nenulovou amplitudou bychom získali korelační koeficient o hodnotě 1. Takové signály jsou tedy v zcela přímé závislosti a není je možné pouze na základě korelace rozlišit.

Využijeme funkce *abs*, kterou jsme již použili u modulace typu 16-QAM v kapitole 4.3.4 a funkce *sum* [11]. Použitím funkce *abs* na pole reálných hodnot dojde k převodu všech záporných hodnot v poli na kladné. Funkcí *sum* získáme součet těchto hodnot.

Když použijeme tyto funkce na skutečný a ideální nosný signál v daném modulačním stavu a podělíme je mezi sebou, tak získáme poměr amplitudy skutečného signálu ku signálu ideálnímu.

Demodulační algoritmus se skládá ze tří vnořených cyklů a tří vnořených podmínek. Pro každý interval vstupního modulačního signálu se vytváří celkem 16 různých ideálních referenčních nosných signálů, které odpovídají daným modulačním stavům. Každý z referenčních signálů je poté porovnán se skutečným modulovaných signálem v da-

ném intervalu modulačního signálu. Výsledkem tohoto porovnávání je zjištění úrovně demodulovaného signálu, pro kterou měl ideální nosný a skutečný modulovaný signál nejvyšší korelační koeficient a blízký poměr ampliudy.



Obrázek 4.5.3: Celková modulace a demodulace 16-QAM

5 Vytvoření návodů pro počítačová cvičení

5.1 Téma příkladů

Pro počítačová cvičení jsem vytvořil jeden rozsáhlejší příklad pro typ modulace 8-QAM. Součástí příkladu je modulace, tvorba konstelačních diagramů, práce se šumem, výpis do příkazového řádku a demodulace.

V příkladech vycházím přímo z modulačních a demodulačních algoritmů modulace typu 16-QAM z kapitol 4.3.4 a 4.5.3 a tvorby konstelačního diagramu pro modulaci 4-QAM z kapitoly 4.3.3. Zveřejněné kódy se liší pouze v názvosloví, ale plní stejnou funkci jako u modulací 4-QAM a 16-QAM. Cílem této kapitoly je vysvětlení vykreslovacích funkcí, které jsou součástí základní sady funkcí GNU Octave [11].

5.2 Tvorba konstelačních diagramů

Nejprve si zkusíme vykreslit různé rozložení modulačních stavů v konstelačním diagramu.

```
c1_8QAM = [1−1j, 1+1j, −1+1j, −1−1j, 3, 3j, −3, −3j];
c2_8QAM = [3, 3+3j, 3j, −3+3j, −3, −3−3j, −3j, 3−3j];
c3_8QAM = [3+1j, 1+3j, −1−3j, −3+1j, −3−1j, −1+3j, 1−3j, 3−1j];
c4_8QAM = [1+1j, 3+3j, −3+3j, −1+1j, −1−1j, −3−3j, 1−1j, 3−3j];
```

Výpis 19: Určení modulačních stavů 8-QAM

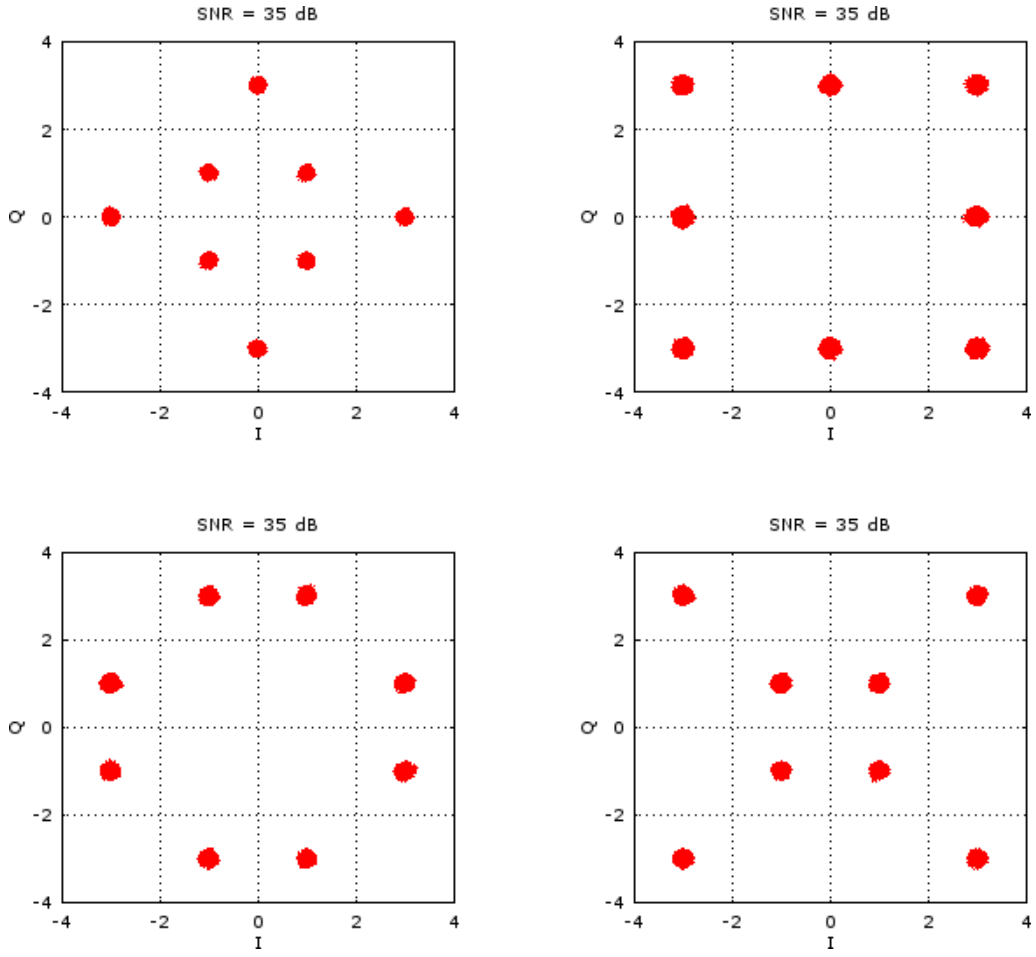
```
figure(8)
subplot(2,2,1);
plot(IQ_8QAM_1, "r*"); title("SNR = 35 dB");
xlabel('I'); ylabel("Q"); grid on

subplot(2,2,2);
plot(IQ_8QAM_2, "r*"); title('SNR = 35 dB');
xlabel('I'); ylabel("Q"); grid on

subplot(2,2,3);
plot(IQ_8QAM_3, "r*"); title('SNR = 35 dB');
xlabel('I'); ylabel("Q"); grid on

subplot(2,2,4);
plot(IQ_8QAM_4, "r*"); title('SNR = 35 dB');
xlabel('I'); ylabel("Q"); grid on
```

Výpis 20: Funkce k vykreslení konstelačních diagramů

Vložení symbolu středník (;) na konec příkazu umožnuje vysázet několik příkazů na jeden řádek ve skriptovacím souboru. Příkaz *figure* vytváří vykreslovací okno, které můžeme vidět na obrázku 3.4.10. Vykreslovací okno je dále rozdělitelné na tabulku o více

dílcích příkazem *subplot*, kde určíme počet řádků, sloupců a index okna tabulky. Příkazem *title* zobrazíme nadpis nad vykresleným obrázkem a příkazy *xlabel* a *ylabel* slouží pro popis os x a y v grafu. Příkaz *grid on* zobrazní v grafu mřížku, která je vhodná pro vykreslování konstelačních diagramů.

Samotná funkce pro vykreslení 2D grafů se provede příkazem *plot*. V nejjednodušší formě má pouze jeden argument a to soubor souřadnic na ose y. Souřadnice jsou na ose x doplněny automaticky a značí indexy v poli argumentu. Dále disponuje velkým množstvím nepovinných argumentů. Pro náš případ dojde k vykreslení červených bodů hvězdicového tvaru.



Obrázek 5.2.1: Vykreslené různé konstelační diagramy

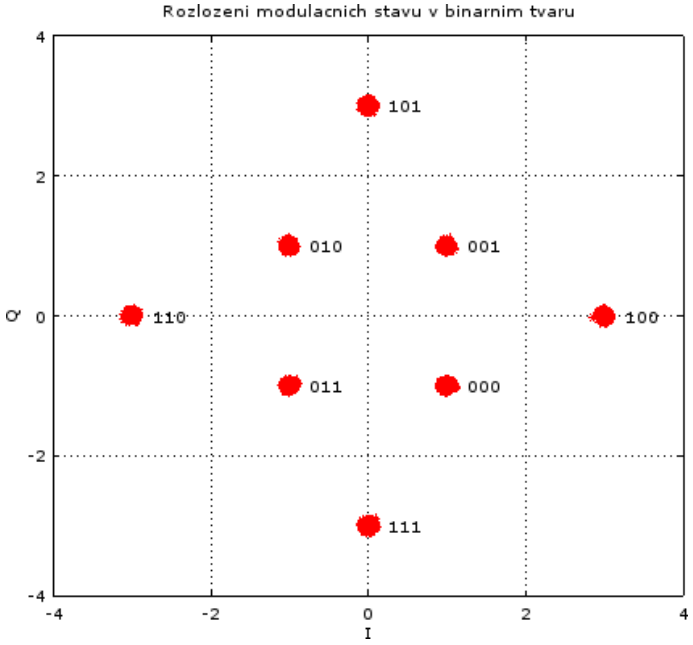
V kapitole 4.1 jsem vysvětlil, že každý z modulačních stavů odpovídá určité hodnotě modulačního signálu. Tato hodnota v desítkové soustavě pak udává, jaká sekvence číšel v binární soustavě je pod danou hodnotou zakódována.

V jednom z vykreslených konstelačních na obrázku 5.2.1 si doplníme označení modulačních stavů a jakou binární kombinaci kódují.

```
figure(9)
plot (IQ_8QAM_1, "r*");
 title ('Rozlozeni modulacnich stavu v binarnim tvaru');
xlabel('I'); ylabel("Q");
x_8QAM = (0:pocet_stavu_8QAM-1);
text(real(c1_8QAM)+0.25, imag(c1_8QAM), dec2bin(x_8QAM))
grid on
```

Výpis 21: Funkce vykreslení modulačních stavů v binární soustavě

Vytvoříme si pole *x_8QAM* v rozmezí hodnot 0–7, které odpovídají počtu modulačních stavů pro 8-QAM. Pro vložení popisků vedle souřadnic modulačních stavů slouží příkaz *text*. Zároveň k argumentu souřadnic použijeme další argument a to funkci *dec2bin* pro vypsání sekvence čísel v binární soustavě, která jen tímto stavem kódovaná.



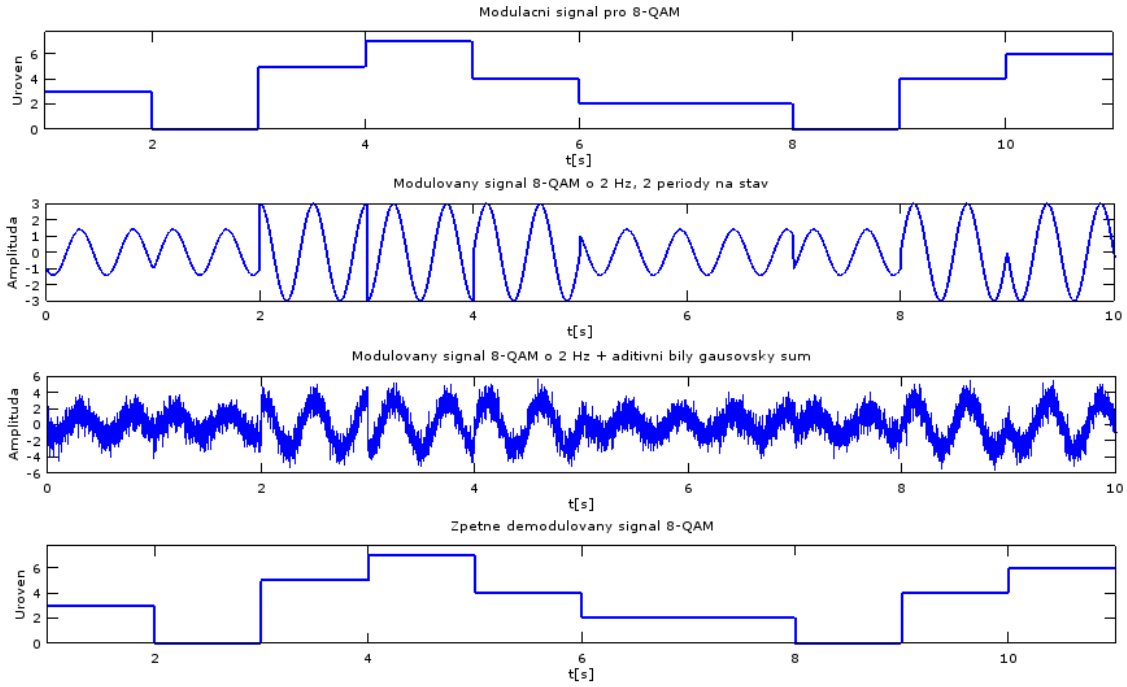
Obrázek 5.2.2: Vykreslený konstelační diagram s označením stavů

5.3 Vykreslení modulace a demodulace

Vybereme si jeden z konstelačních diagramů pro 8-QAM a vykreslíme jeho modulaci a demodulaci. Rozložení modulačních stavů v kompexní rovině vychází z hodnot v poli *c1_8QAM*.

```
figure(10)
subplot(4,1,1);
stairs(modulacni_signal_8QAM, 'LineWidth', 2);
axis([1 pocet_intervalu+1 0 pocet_stavu_8QAM-0.2])
 title ('Modulacni signal pro 8-QAM');
xlabel('t[s]'); ylabel("Uroven");

subplot(4,1,2)
plot(t, modulovany_signal_8QAM, 'LineWidth', 2);
xlim([0 pocet_intervalu]);
xlabel('t[s]'); ylabel("Amplituda");

subplot(4,1,3)
plot(t, modulovany_signal_8QAM_S, 'LineWidth', 1);
xlim([0 pocet_intervalu]);
 title ('Modulovany signal 8-QAM o 2 Hz + aditivni bily gausovsky sum');
xlabel('t[s]'); ylabel("Amplituda");

subplot(4,1,4);
stairs(demodulovany_signal_8QAM_S, 'LineWidth', 2);
axis([1 pocet_intervalu+1 0 pocet_stavu_8QAM-0.2])
 title ('Zpetne demodulovany signal 8-QAM');
xlabel('t[s]'); ylabel("Uroven");
```

Výpis 22: Funkce vykreslení modulace a demodulace



Obrázek 5.3.1: Vykreslená modulace a demodulace

K vykreslení vstupního modulačního signálu použijeme funkci *stairs*, jelikož se jedná o signál obdélníkového průběhu. Funkce vykresluje začátek signálu na pozici 1 namísto 0 a právě proto nejsou průběhy obdélníkových a harmonických signálů zarovnány přesně pod sebou a jsou vychýlené o jednu hodnotu na ose x. Abychom toto vykompenzovali, použijeme příkaz *axis* a do jeho argumentu vložíme vektor rozmezí hodnot osy x a y.

Pro všechny signály nastavíme v argumentu funkcí *stairs* a *plot* dvojnásobnou šířku čáry a pouze u modulovaného signálu s přičteným šumem ji ponecháme v základní šířce. Při použití funkce *plot* se nastavuje rozsah hodnot na ose x a y dvěma příkazy, *xlim* a *ylim*.

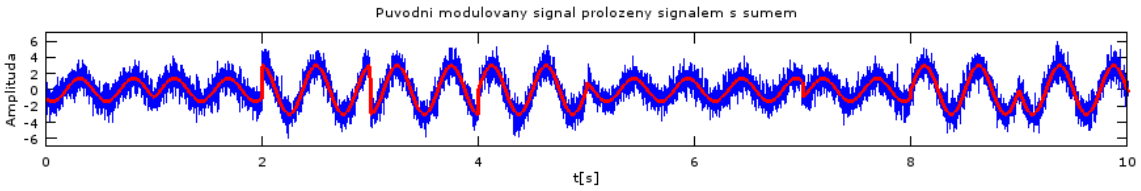
Jak je možné si všimnout, tak všechny argumenty příkazů, které určují rozsahy hodnot na osách x a y, jsou hlavní proměnné k nastavení simulace. Nemusíme tedy jakkoliv upracovat vykreslovací funkce, když se rozhodneme změnit parametry simulace.

GNU Octave dovoluje zobrazit více hodnot pole do jednoho grafu. Zobrazíme si proložení ideálního a skutečného modulovaného signálu pro 8-QAM.

```
figure(11)
plot(t, modulovany_signal_8QAM_S, 'LineWidth', 1);
hold on;
plot(t, modulovany_signal_8QAM, 'LineWidth', 3, "r");
xlim([0 pocet_intervalu]); ylim([-7 7]);
 title ('Puvodni modulovany signal prolozeny signalem s sumem');
xlabel('t[s]'); ylabel("Amplituda");
```

Výpis 23: Funkce vykreslení proložených signálů

V rámci celého vykreslovacího okna *figure* či tabulky *subplot* můžeme použít vykreslovací funkce jako *plot* vícekrát. Slouží k tomu příkaz *hold on*. Vykreslíme si ideální modulovaný signál pro 8-QAM červenou barvou a proložíme jej skutečným modulovaným signálem s hodnotou SNR o 30 dB a to v základní modré barvě.



Obrázek 5.3.2: Vykreslení proloženého ideálního a skutečného modulovaného signálu

5.4 Výpis do příkazového řádku

Když provádíme simulaci modulace a demodulace a kontrolujeme správnost dat na výstupu, je vhodné průběžně vypisovat klíčové proměnné. Tento výpis můžeme provádět do příkazového řádku GNU Octave, jelikož nám program umožnujě i práci s textem.

```
printf("Modulacni signal: %g\n", modulacni_signal_8QAM(i))
printf("Modulacni stav: %d %+d j\n", real(c1_8QAM(modulacni_signal_8QAM(i)+1)), imag(
    ↪ c1_8QAM(modulacni_signal_8QAM(i)+1)))
printf("Amplituda: %g\n", abs(c1_8QAM(modulacni_signal_8QAM(i)+1)))
printf("Uhel pocatecni faze v radianech: %g\n", angle(c1_8QAM(modulacni_signal_8QAM(i)+1)))
printf("Uhel pocatecni faze ve stupnich: %g\n", rad2deg(angle(c1_8QAM(
    ↪ modulacni_signal_8QAM(i)+1))))
printf("\n")
```

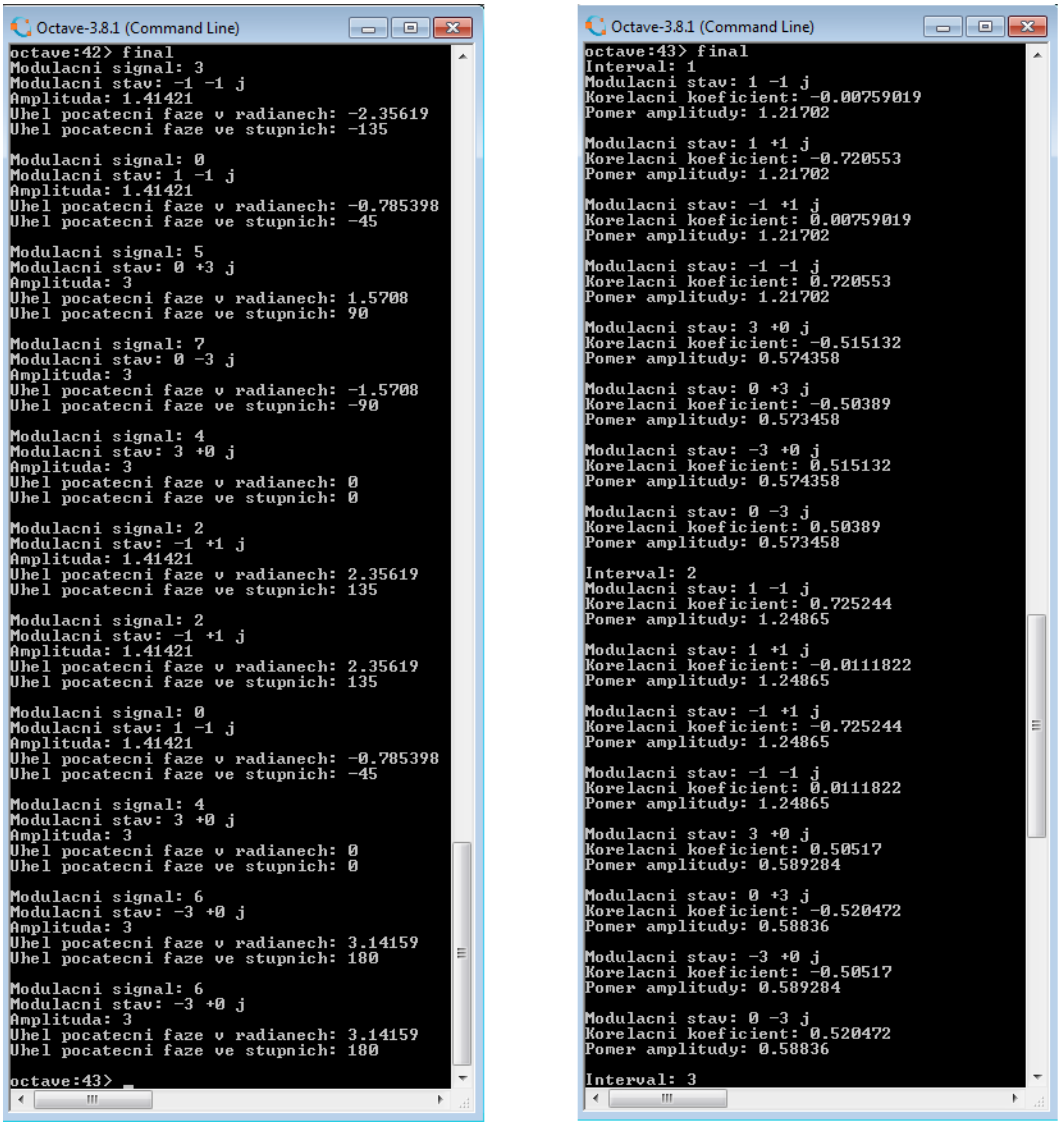
Výpis 24: Funkce pro výpis klíčových proměných při modulaci

```
printf("Interval: %g\n", i)
printf("Modulacni stav: %d %+d j\n", real(c1_8QAM(p)), imag(c1_8QAM(p)))
printf("Korelacni koeficient: %g\n", corr(modulovany_ve_stavu_8QAM, referencni_nosna_8QAM)
    ↪ )
printf("Pomer amplitudy: %g\n", sum(abs(modulovany_ve_stavu_8QAM)) / sum(abs(
    ↪ referencni_nosna_8QAM)))
printf("\n");
```

Výpis 25: Funkce pro výpis klíčových proměných při demodulaci

Příkazy ve výpisu 24 jsou součástní modulačního algoritmu a příkazový výpis 25 vložíme do algoritmu demodulace. Když nechceme vypisovat hodnoty proměnných do příkazového řádku, tak tyto příkazy označníme jako komentář vložením symbolu % na začátek řádku. Příkazy pro průběžný výpis klíčových proměnných modulace a demodudace poté nebudou provedeny.

Formátovaný výstup se provádí příkazem *printf*. První argument určuje zobrazený text a umístění proměnných zvolitelného datového typu. Do druhého argumentu se vkládá název proměnných pro vypsání jejich hodnot. Mimo funkcí ze základní sady jsem použil i funkci *rad2deg* [13], která převádí radiány na stupně.



Obrázek 5.4.1: Výpis klíčových proměnných modulace a demodulace

Zobrazený výpis klíčových hodnot se vztahuje k modulace a demodulaci pro 8-QAM na obrázku 5.3.1.

Můžeme například vidět, že při modulačním signálu o hodnotě 2, kterému odpovídá modulační stav na souřadnicích $-1+1j$ došlo k modulaci nosným signálem o amplitudě $1,41421$ a počáteční fázi rovné $2,35619$ rad neboli $135°$.

Z výpisu demodulace můžeme zjistit, že v prvním testovaném intervalu modulačního signálu dosáhl čtvrtý modulační stav na souřadnicích $-1-1j$ nejvyššího korelačního koeficientu o hodnotě $0,720553$ a poměru amplitudy skutečného ku ideálnímu modulovanému signálu o hodnotě $1,21702$. Úroveň výsledného demodulovaného signálu je tedy rovna hodnotě 3.

6 Závěr

Cílem bakalářské práce bylo vytvořit simulaci ASK, PSK, FSK a QAM modulace a demodulace digitálních signálů s využitím programu GNU Octave. Tvorba simulací vychází z teoretického výkladu modulačních a demodulačních principů a matematických rovnic použitých signálů.

Simulaci jsem provedl v programu GNU Octave. Vypsal jsem jeho základní vlastnosti, výhody a nevýhody a seznam balíčků funkcí, které jsem při simulaci využil. Zvolil jsem verzi programu a popsal jsem průběh instalace a úvodní konfigurace pod operačním systémem Windows 7, x64.

V samotné realizaci simulace jsem nejprve modulovoval konkrétní typy jednoduchých dvoustavových a složených vícestavových modulací a dále vytvářel modulační schémata, tzv. konstelační diagramy. Na základě teoretických předpokladů jsem ověřil chování signálu, který je ovlivněn působením šumu typu AWGN. Pro ruzné hodnoty SNR jsem signály demoduloval. Proces modulace a demodulace jsem prováděl ve vlastních neprve jednodušších a poté složitějších zautomatizovaných algoritmech. Řešené příklady jsem doplnil vysvětleným programovým kódem a grafickým výstupem.

Při tvorbě návodů ke cvičení jsem se blíže zabýval vysvětlením používání vykreslovacích funkcí. Součástí návodů je i formátovaný výstup hodnot klíčových proměnných do příkazového řádku pro snažší orientaci a ověření správnosti výsledných hodnot modulace a demodulace.

Výsledkem práce je vlastní programový kód o 545 řádcích uložený ve formě skriptovacího souboru. Jeho obsahem je celková simulace ASK, FSK, PSK, 4-QAM, 8-QAM a 16-QAM modulace a demodulace digitálních signálů. Dále obsahuje tvorbu konstelačních diagramů a výpis formátovaného výstupu. Programový kód je možné volně přenášet, modifikovat či vylepšovat a může sloužit jako výukový nástroj pro studenty, kteří se věnují problematice modulací a demodulací. Korektní spuštění skriptovacího souboru jsem otestoval na programu GNU Octave verze 3.8.1 s využitím balíčků funkcí communications 1.2.0, control 2.6.6, general 1.3.4, nurbs 1.3.9 a signal 1.3.0. Skriptovací soubor jsem na závěr umístil do archívu s názvem přílohy.zip a uložil jej na přiložené DVD.

7 Reference

[1] MACHÁČEK, Zdeněk a Pavel NEVŘIVA. *Modulované signály.* Vyd. 1. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2012. ISBN 978-80-248-2600-4.

[2] ŠEBESTA, Vladimír a Zdeněk SMÉKAL. *Signály a soustavy: přednášky.* Vyd. 1. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2003, 145 s. ISBN 80-214-2434-6.

[3] Jareš, Petr. *Moderní modulační metody a jejich aplikace.* [online]. České vysoké učení technické v Praze. [cit. 2015-04-15]. Dostupný z WWW: <http://data.cedupoint.cz/oppa_e-learning/2_KME/163.pdf>.

[4] SCHMIDT HANSEN, Jesper. *GNU Octave: beginner's guide : become a proficient Octave user by learning this high-level scientific numerical tool from the ground up.* Birmingham: Packt Pub., 2011, vii, 258 p. ISBN 978-1-849513-32-6.

[5] Richterek, Lukáš. *Dynamické modelování v programu GNU Octave.* [online]. Katedra experimentální fyziky PřF UP Olomouc, poslední revize 18.10.2007 [cit. 2015-04-15]. Dostupný z WWW: <http://muj.optol.cz/richterek/lib/exe/fetch.php?media=texty:dynmod.pdf>.

[6] David Bateman, Laurent Mazet and Paul Kienzle. *Communications Toolbox for Octave.* [online]. last revision March 2003 [cit. 2015-04-15]. Dostupný z WWW: <http://www.roguehangar.com/mathpages/SRIP_docs/MatLabWork/Octave/share/octave/packages/communications-1.0.4/doc/comms.pdf>.

[7] Just, Michal. *Octave - český průvodce programem.* [online]. poslední revize červen 2006 [cit. 2015-04-15]. Dostupný z WWW: <http://www.octave.cz/>.

[8] Stopka, Marek. *GNU Octave.* [online]. poslední revize 20.2.2013 [cit. 2015-04-15]. Dostupný z WWW: <http://www.abclinuxu.cz/software/veda/matematika/gnu-octave>.

[9] *Domovská stránka GNU Octave.* [online]. [cit. 2015-04-15]. Dostupný z WWW: <http://www.gnu.org/software/octave/>.

[10] *Stránka archívu novinek GNU Octave verze 3.8.* [online]. [cit. 2015-04-15]. Dostupný z WWW: <https://www.gnu.org/software/octave/NEWS-3.8.html>.

[11] *Stránka Octave-Forge : základní sada funkcí GNU Octave* [online]. [cit. 2015-04-15]. Dostupný z WWW: <http://octave.sourceforge.net/octave/overview.html>.

[12] *Stránka Octave-Forge : instalovatelné balíčky funkcí* [online]. [cit. 2015-04-15]. Dostupný z WWW: <http://octave.sourceforge.net/packages.php>.

[13] *Stránka Octave-Forge : seznam všech funkcí možných balíčků* [online]. [cit. 2015-04-15]. Dostupný z WWW: <http://octave.sourceforge.net/functions_by_package.php>.

[14] Weisstein, Eric W. *Complex Number.* [online]. From MathWorld–A Wolfram Web Resource. [cit. 2015-04-15]. Dostupný z WWW: <http://mathworld.wolfram.com/ComplexNumber.html>.

[15] DOBEŠ, Josef a Václav ŽALUD. *Moderní radiotechnika.* 1. vyd. Praha: BEN - technická literatura, 2006, 767 s. ISBN 80-7300-132-2.